Серия "Учебно-методический комплекс дисциплины"

МУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет математики и информатики

Т.А. Галаган

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ JAVASCRIPT

Практикум

Федеральное агентство по образованию
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Серия «Учебно-методический комплекс дисциплины»

Т.А. Галаган

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ JAVASCRIPT

Практикум

Благовещенск
2008

ББК 32.973 - 018.1я73
П 78

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Галаган Т. А. (*составитель*)

Программирование на языке JavaScript. Практикум. Для студентов специальностей 230102 «Автоматизированные системы обработки информации и управления» и 230201 «Информационные системы в технологии» очной формы обучения. – Благовещенск: Амурский гос. ун-т, 2008.

Пособие посвящено основам программирования на JavaScript, позволяющего создавать интерактивные web-приложения. Рассмотрены основные элементы языка, приведены примеры программ, варианты индивидуальных заданий.

Рецензенты: Попова Е.Ф., *доц. кафедры информатики БГПУ, канд. техн. наук*
Макарчук Т.А., *доц. каф. общей математики и информатики АмГУ, канд. пед. наук*

ВВЕДЕНИЕ

JavaScript – язык для составления сценариев, позволяющий легко создавать интерактивные web-страницы, содержащие динамические модули.

Конструкции языка JavaScript очень схожи с операторами языка C++, но имеют свои особенности. Их освоение не составит труда как для человека, знающего основы программирования на любом другом языке, так и для новичка.

Для создания сценариев на JavaScript требуется знание основ другого языка – HTML. Это язык разметки гипертекстовых документов, именно с его использованием реализуется создание web-страниц. Поэтому в приложении данного пособия содержатся основные теги и справочная информация языка HTML.

Практикум предназначен для студентов третьего курса, уже имеющих навыки программирования. Он содержит набор лабораторных работ, в каждой из которых приводятся необходимый теоретический материал, примеры, варианты заданий для самостоятельного выполнения, контрольные вопросы.

Тематика работ посвящена не только изучению синтаксических конструкций языка, но и методам динамического преобразования HTML-документов.

Пособие служит дополнительным материалом при проведении лабораторных работ по дисциплинам «Мировые информационные ресурсы и сети» специальности 230201 и «Сети ЭВМ и телекоммуникации» специальности 230102.

Лабораторная работа №1

Основы JavaScript

Для выполнения программ JavaScript в качестве интерпретатора (исполнительной системы) можно взять веб-браузер Internet Explorer 6.0, в качестве редактора программ – текстовой редактор, – например, Блокнот. Можно создавать программы на JavaScript и с помощью специальных программ, предназначенных для разработки веб-страниц, - например, Microsoft FrontPage.

Можно легко создать и собственный редактор программ, – например, кодом:

```
<HTML>
<H3>Редактор кодов</H3>
код:<br>
<TEXTAREA id="mycode" ROWS=10 COLS=60></TEXTAREA>
<p>Результат:<br>
<TEXTAREA id="myrezult" ROWS=3 COLS=60></TEXTAREA>
<P>
<BUTTON onclick="document.all.myrezult.value=eval(mycode.value)">
Выполнить</BUTTON>
<BUTTON onclick="document.all.mycode.value='';
document.all.myrezult.value=''">
Очистить </BUTTON>
<P>
<!Комментарий>
Введите текст программы в верхнее окно.
</HTML>
```

Ввод-вывод данных

Для обеспечения ввода-вывода JavaScript предоставляет несколько методов: alert( ), confirm ( ), prompt( ).

Первый из перечисленных выводит на экран диалоговое окно с заданным сообщением и кнопкой ОК (рис.1). Синтаксис данного метода:

alert (сообщение)

Сообщение может представлять собой данные любого типа: последовательность символов, заключенную в кавычки, число, переменную или выражение. Текст необходимо заключить в кавычки. Например, alert ("Hello, JavaScript !")

Рис.1. Диалоговое окно метода alert( )

Рис.2. Диалоговое окно метода confirm( )

Окно, создаваемое alert( ), является модальным (останавливающим все последующие действия программы и пользователя). Его можно убрать, щелкнув по кнопке ОК.

Метод confirm выводит на экран диалоговое окно с сообщением и двумя кнопками – ОК и Отмена (рис.2). Этот метод возвращает логическую величину, значение которой зависит от того, по какой из кнопок щелкнет пользователь. Возвращаемое значение можно обработать в программе, создавая тем самым интерактивный эффект. Синтаксис применения данного метода аналогичен синтаксису метода alert. Окно, создаваемое confirm, также является модальным.

Метод prompt осуществляет вывод диалогового окна с сообщением и кнопками ОК и Отмена, а также с текстовым полем, в которое пользователь может ввести данные (рис.3).

Рис.3. Диалоговое окно метода prompt( )

В отличие от alert( ) и confirm( ) данный метод имеет два параметра: текст сообщения и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет по кнопке ОК, метод вернет содержимое поле ввода данных, если по кнопке Отмена, то возвращается значение ложь. Возвращаемое значение можно также обработать в программе. Синтаксис метода:

prompt (сообщение, значение_поля_ввода данных)

Оба параметра не являются обязательными. Если они не указаны, на экране появится окно без сообщения, а в поле ввода данных подставлено значение по умолчанию – undefined (не определено). Чтобы значение по умолчанию не появилось, в качестве второго параметра указывается пустая строка (" ").

y=prompt( )

Типы данных

Типы данных языка JavaScript приведены табл. 1.

Типы данных *Таблица 1*

Тип данных	*Описание значений*
Строковый или символьный тип (string)	Последовательность символов, заключенная в кавычки, двойные или одинарные
Числовой (number)	Число положительное или отрицательное, последовательность цифр. Целая и дробная части разделяются точкой
Логический	Два значения: true или false
Null	Отсутствие какого-либо значения
Объект (object)	Программный объект с собственными свойствами. В частности массив также является объектом.
Функция (function)	Программный код, выполнение которого может возвращать некоторое значение

При создании программ за типом данных следит программист. Интерпретатор не выдаст ошибки при неверном их использовании. Он просто попытается привести данные к типу, требуемому в данной операции.

Например, при написании выражения 7+ "нет" результатом будет строка символов "7нет". Интерпретатор сначала переводит число в строку, а затем выполняет сложение двух строк, результатом которого в JavaScript является сложение двух строк. Результатом вычисления выражения 5+6 будет 11, а выражения 5+"6" – "56".Для преобразования строк в числа предусмотрены встроенные функции parseInt( ) и parseFloat( ), каждая из которых имеет два параметра: первый – строка, второй – основание.

Если основание не указано, то предполагается 10 – десятеричная система счисления. В качестве основания можно также использовать 8 и 16.

При преобразовании строки в целое число округления не происходит – дробная часть просто отбрасывается.

Примеры

```
parseInt( "8.94")       // результат 8
parseFloat("8.94")      // результат 8.94
parseInt( "весна" )     // результат NaN – значение не определено
parseFloat( "весна" )   // результат NaN – значение не определено
parseInt( "15", 8 )     // результат 255
parseFloat( "17.5")     // результат 435
```

Переменные

Имя переменной представляет собой конечную последовательность символов, содержащую буквы, цифры, символ подчеркивания. Имя переменной не должно начинаться с цифры или содержать пробелы. Для имен переменных нельзя использовать ключевые слова языка.

В отличие от многих других языков программирования переменной не нужно задавать тип при объявлении. Тип переменной определяется типом ее значе-

ния. Переменная может принимать значения разных типов и неоднократно его изменять.

Создавать переменную в программе можно несколькими способами. Можно просто присвоить ей значение с помощью оператора присваивания в формате:

имя_переменной = значение

Например,

Month= "Январь"

Возможно использование ключевого слово var перед именем переменной. В этом случае переменная не будет иметь первоначального значения, но в дальнейшем его можно передать с помощью оператора присваивания. Например,

var Month

Month = "Январь"

При использовании var допускается и инициализация переменной, например:

var Month = "Январь"

Для каждой переменной инициализация при объявлении возможна только один раз.

Можно сразу объявить несколько переменных, используя var и разделяя их запятой, при этом возможно инициализировать их все или некоторые из них:

var Month= "Январь", day, pi=3.14, x

Операции

В табл. 2 заданы основные операции, определенные в языке JavaScript. Арифметические операции могут применяться к данным любых типов. В их обычном понимании они применяются к числам. В случае использования строковых данных операция сложения реализуется конкатенацией (склеиванием строк). Применение остальных операций к строкам дает в результате NaN. При выполнении арифметических операций с логическими данными интерпретатор переводит логические значения в числовые (true в 1, false в 0) и возвращает числовой результат.

Применение арифметических операций к разнотипным значениям может привести к непредсказуемым результатам.

Операции JavaScript *Таблица 2*

Операция	*Краткое описание*
Унарные операции	
++	инкремент (увеличение на 1)
--	декремент (уменьшение на 1)
Бинарные операции	
*	умножение
/	деление
%	остаток от целочисленного деления
+	сложение
-	вычитание
<	меньше
<=	меньше или равно
>=	больше или равно
= =	равно
!	отрицание (не)
&&	логическое И
||	логическое ИЛИ
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием
%=	остаток от деления с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием

Операции выполняются в соответствии с приоритетами. Приоритет опера-

ций аналогичен языку C++. В табл. 2 операции расположены в порядке убывания или равенства приоритета. Для изменения порядка выполнения операций используются круглые скобки.

Примеры:

var a=1, b=3, d=10

d+= 4*(a + 9/b)

Здесь значение переменной d равно 26.

Комментарии

В JavaScript допустимы два вида операторов комментария: одна строка символов, расположенная справа от символов // или произвольное количество строк, заключенных между символами /* и */.

Операторы ветвления

Операторы ветвления также сохраняют преемственность языка C++.

Условный оператор if

Условный оператор if используется для разветвления процесса вычислений на два направления. Синтаксис оператора if:

if (условие) оператор1 else оператор 2

Например:

if (fvalue>=0.0) fvalue = fvalue else fvalue = -fvalue

// вычисляется модуль произвольного числа.

Ветвь с ключевым словом else может отсутствовать. Например:

if(f != 0) l = 100 / f

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок (операторные скобки { }), иначе компилятор не сможет определить окончание ветвления.

if (a) { a++ v=60*a } else v = a

if (e ==1000) { e /=10 alert(e) } else { e =10+y*y y++ }

Условные операторы могут быть вложенными.

if (a<b) { if (a<c) m = a else m = c } else if (b<c) m = b else m = c

Необходимо помнить, что в этом случае else относится к ближайшему if. Операторные скобки после первого if необязательны.

Если требуется проверить несколько условий, их объединяют знакам логических операций.

Оператор выбора switch

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Синтаксис оператора:

switch (выражение) {

case константное выражение 1 : операторы1 break

case константное выражение 2 : операторы2 break

...

case константное выражение n : операторыN break

default : операторы }

Выполнение оператора начинается с вычисления выражения, а затем управление передается первому оператору из списка, помеченному константным выражением, значение которого совпало с вычисленным.

После этого, если выход из переключателя явно не указан (отсутствует break), последовательно выполняются все нижележащие ветви. Выход из переключателя обычно выполняется с помощью оператора break.

Если совпадения не произошло, выполняются операторы, расположенные после ключевого слова default. Ветвь default может отсутствовать.

Пример. Печать названия месяца по порядковому номеру

```
switch (x ) {
    case 1 : alert( "Январь") ;      break
    case 2 : alert( "Февраль");      break
    case 3 : alert( "Март");         break
    ...
```

```
        case 12 : alert( "Декабрь");    break
        default  : alert( "Неверный номер" )
    }
```

Параметр выражения оператора switch может принимать строковые, числовые и логические значения. В следующем примере переменная x содержит название языка, который выбрал пользователь. А выражение window.open ( ) открывает новое окно браузера и загружает в него указанный в скобках HTML-документ.

```
switch (x ) {
    case " английский" : window.open ( "engl.htm") ;        break
    case " французский" : window.open ( "french.htm") ; break
    case " русский" :  window.open ( "russ.htm") ;          break
    default : alert( "Нет документа на таком языке" )
}
```

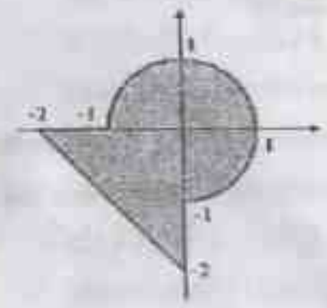
Задание

1. Создать собственный редактор программ, на основе HTML-кода, модифицировав код, приведенный в тексте. Изменить его размеры, цвет фона, комментарии. Файл открыть в браузере как веб-страницу.

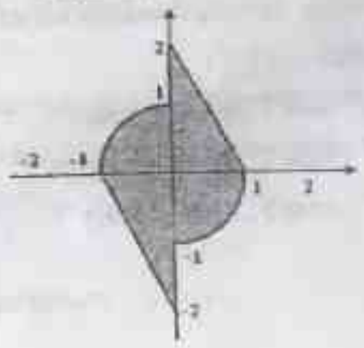
2. Протестировать редактор, изучив в нем функции вывода и основные операции JavaScript.

3. Составить программу на основе разветвляющего алгоритма для задачи: задана «мишень» в виде закрашенной области, изображенной на рисунке. Создать программу, определяющую, попадает ли в нее точка с координатами (x, y).

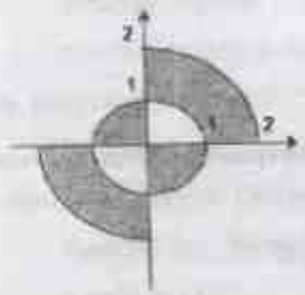
Вариант 1



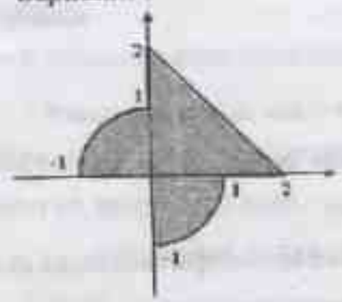
Вариант 2



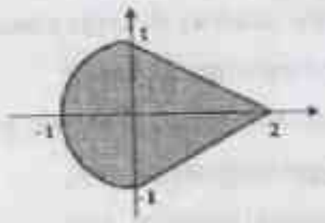
Вариант 3



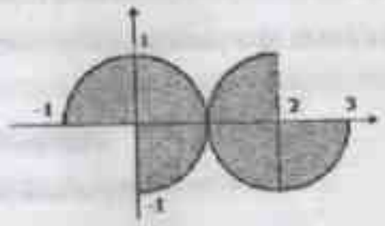
Вариант 4



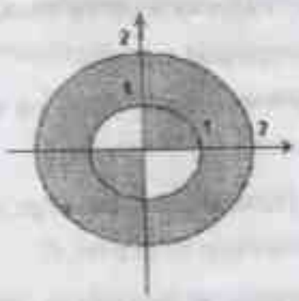
Вариант 5



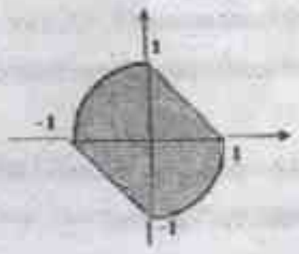
Вариант 6



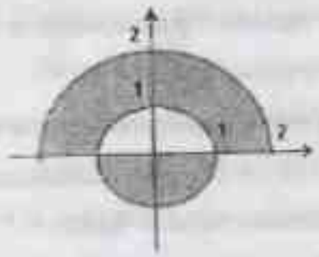
Вариант 7



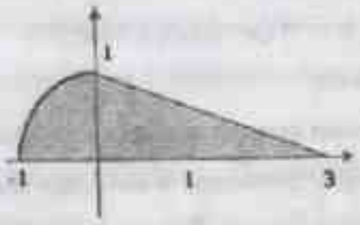
Вариант 8



Вариант 9



Вариант 10



Контрольные вопросы

1. Что такое модальное окно?

2. Каков синтаксис метода prompt? Какие из его параметров являются необязательными?

3. Какие операторы ветвления существуют в JavaScript?

4. Перечислите основные типы данных JavaScript.

5. В чем отличие объявление переменных в JavaScript от объявлений переменных в языках программирования высокого уровня?

Лабораторная работа №2

Функции и операторы цикла

JavaScript содержит следующие встроенные функции:

parseInt(строка, основание) преобразует указанную строку в целое число по указанному основанию(8, 10, 16); по умолчанию – десятеричная система.

parseFloat(строка, основание) преобразует строку в число с плавающей точкой.

isNaN(строка, основание) возвращает true, если указанное в параметре значение не является числом.

eval(строка) вычисляет выражение в указанной строке, выражение не должно содержать тегов HTML. Например:

```
var a = 10;                          // значение a равно 10
var b = "if (a<=25) { a *= 2 }"      // значение b равно строке символов
eval (b)                             // значение a равно 20
```

Другой пример применения данной функции в тексте программы редактора, где сценарии записаны в качестве значений атрибутов onclik, определяющих событие щелчок кнопкой мыши на HTML-кнопках, заданных тегами <button>.

escape(строка) возвращает строку в виде %XX, где XX – ASCII-код указан-

ного символа. Такую строку называют escape-последовательностью.

unescape(строка) – обратное преобразование.

type0f (объект) возвращает тип указанного объекта в виде символьной строки, например, "boolean", "function".

В программах на JavaScript пользователям также разрешено создавать собственные функции. Объявление функция состоит из заголовка и тела:

```
function имя_функции ( параметры )          //заголовок функции
{ тело функции }
```

Тело функции заключается в фигурные скобки. Параметры функции, стоящие в круглых скобках, перечисляются через запятую.

Возвращение значений из функции происходит с помощью оператора return, за которым помещается само возвращаемое значение.

Для вызова функции можно воспользоваться выражениями вида:

имя_функции (параметры) или

имя_переменной = имя_функции (параметры)

Параметры в вызове функции должны быть представлены конкретными значениями.

Например, если описание функции дано в виде:

function cbe (x) { return x*x*x }

ее вызов может быть записан в виде: y=cbe(25)

В JavaScript функции могут вызываться как после их определения, так и до него. Можно не поддерживать соответствие количества параметров в определении функции и в ее вызове. Если в определение функции параметров больше, чем в вызове, то недостающим параметрам автоматически присваивается значение null. Лишние параметры в вызове функции игнорируются.

Внутри функции можно создавать переменные с помощью оператора присваивания или с помощью ключевого слова var.

Если в теле функции переменная в составе оператора присваивания встречается впервые в программе или была определена до этого – она действует как глобальная.

Если в теле функции используется переменная, объявленная только во внешней программе, она также является глобальной.

Если для определения переменной в теле функции используется ключевое слово var, она будет локальной вне зависимости от того, определена она во внешней программе или нет.

Операторы цикла

Как и другие языки программирования JavaScript имеет три вида оператора цикла: цикл с предусловием (while), цикл с постусловием (do while), цикл с параметром (for).

Синтаксис цикла с предусловием:

while (условие) { операторы }

Выражение, стоящее в круглых скобках, определяет условие повторения тела цикла, представленного простым или составным оператором. Если оператор простой операторные скобки { } могут не ставиться.

Выполнение оператора цикла начинается с вычисления выражения. Если оно истинно, выполняется тело цикла. Если при первой проверке выражение ложно (false), цикл не выполнится ни разу.

```
// Программа вычисляет возведение 10 в степень 5.
var x =10
y=2
while ( y <= 5 )   {  x*=10; y++  }
```

Цикл while обычно используется в тех случаях, когда число повторений заранее не известно. В этом случае используется и оператор цикла с постусловием. Его отличительной чертой является выполнение тела цикла хотя бы один раз. И только после первого его выполнения проверяется, надо ли его выполнять еще раз. Таким образом, даже если условие заведомо ложно цикл выполнится один раз. Если условие истинно, тело цикло выполнится еще раз. Цикл завершается, когда выражение станет равным false или в теле цикла будет выполнен оператор передачи управления.

Цикл for называют также циклом с заданным числом повторений. Он имеет

следующий формат:

for (инициализация, выражение (условие); модификации)

{операторы }

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. Инициализация выполняется один раз, перед выполнением тела цикла.

Выражение определяет условие выполнения цикла: если его результат равен истине, то цикл выполняется. Цикл с параметром реализуется также как цикл с предусловием.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла.

Тело цикла представляет собой простой или составной оператор.

```
for (i = 1, s=1; i<11; i++) s*=i;                    // вычисления факториала 10
```

Для принудительного выхода из тела любого цикла используются операторы break и continue. Первый позволяет переход в точку программы, находящуюся непосредственно за оператором, внутри которого находится, т.е. управление передается первой строке, следующей за телом цикла.

```
time=1; sum=0
while (time<10)
{ sum+=time
      if ( sum>20 )       break;
 time+=   }
```

Инструкция continue заставляет программу пропустить все оставшиеся строки цикла, но сам цикл при этом не завершается. Для решения некоторых задач удобно комбинировать инструкции break и continue.

Задание

1. Составить программу на основе циклического алгоритма для вычисления суммы ряда с заданной точностью ε. Определите и выведите на экран значение суммы и число элементов ряда, вошедших в сумму.

№ варианта	Задание	Ограничения	Точность	Результат
1	$\frac{\pi}{3}+\sum_{n=1}^{\infty}(-1)^n\frac{(\pi/3)^{2n+1}}{(2n+1)!}$	$\lvert x\rvert<\infty$	$0.5*10^{-4}$	$\sin(\pi/3)$
2	$\sum_{n=0}^{\infty}(-1)^n\frac{(\pi/6)^{2n}}{(2n)!}$	$\lvert x\rvert<\infty$	$0.5*10^{-4}$	$\cos(\pi/6)$
3	$\sum_{n=0}^{\infty}\frac{(-1)^n(x-1)^{n+1}}{(n+1)}$	$0<x\le 2$	$0.5*10^{-3}$	$\ln x$
4	$\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}$	$\lvert x\rvert<\infty$	10^{-4}	e^{-x^2}
5	$\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}$	$\lvert x\rvert<\infty$	$0.5*10^{-5}$	$\frac{\sin x}{x}$
6	$\sum_{n=0}^{\infty}\frac{(-1)^{n-1}x^{n+1}}{n+1}$	$\lvert x\rvert<1$	$0.2*10^{-4}$	$\ln(1+x)$
7	$\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{2n+1}$	$\lvert x\rvert<1$	10^{-6}	$arctg\,x$
8	$\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}$	$\lvert x\rvert<\infty$	$0.5*10^{-5}$	$\cos x$
9	$\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}$	$\lvert x\rvert<\infty$	10^{-6}	e^{-x}
10	$2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}$	$x>0$	10^{-4}	$\ln x$

2. Создать пользовательскую функцию, продемонстрировать ее работу в программе.

Вариант 1. Составить программу вычисления значения $a + b$, где

$$a=\frac{\sqrt{z^3+4z^2+7z+1}}{2+e^{z+3}\cdot z-1}, \quad b=\frac{t^2+13t+16}{7t^3+t^2-4}.$$

Для определения значений многочленов использовать функцию.

Вариант 2. Вычислить значение

$$f=\frac{\max(a,b,c)*\min(a,c,d)-\max(b,c,d)}{\min(a,b,c)},$$

где a, b, c, d – некоторые значения, введенные с клавиатуры. Для нахождения максимального и минимального значений использовать функции.

Вариант 3. Создать функцию вывода на экран таблицы умножения для числа, являющегося параметром функции. Продемонстрировать работу функции в программе.

Вариант 4. Создать функцию вычисления площади треугольника по трем заданным сторонам. Продемонстрировать работу функции.

Вариант 5. Создать функцию, определяющую, принадлежит ли точка с заданными координатами (x_1, y_1) уравнению прямой $y = kx + b$. Координаты точки и коэффициенты уравнения прямой объявить параметрами функции. Использовать функцию для определения из набора точек, лежащих на одной прямой.

Вариант 6. Определить среднее арифметическое сторон двух треугольников, заданных координатами их вершин. Длину стороны треугольника вычислять в функции.

Вариант 7. Вычислить среднее арифметическое объемов трех прямоугольных пирамид. Объем пирамиды вычислить в функции, параметрами которой являются длина стороны основания и высота.

Вариант 8. Вычислить значение $(a! + c!)/(a+c)!$, где a и c ввести с клавиатуры. Для вычисления факториала использовать функцию.

Вариант 9. Для каждого из трех заданных радиусом основания и высотой

конусов сфер определить объем.

Вариант 10. Создать функцию, находящую произведение корней квадратного уравнения. Коэффициенты уравнения объявить параметрами функции.

Контрольные вопросы

1.Какие виды операторов цикла существуют в JavaScript? В чем их отличие?

2. Поясните действия инструкций break, return, continue.

3. Как создавать пользовательскую функцию?

4. Назовите особенности вызова функций в JavaScript.

5. Что такое параметры функции?

Лабораторная работа №3

Встроенные объекты JavaScript. Строки и массивы

Объекты представляют собой программные единицы. Как любой объект, объект JavaScript обладает данными (свойствами) и методами (функциями) для их обработки. Управление web-страницами с помощью сценариев, созданных на основе JavaScript, заключается в использовании и изменении свойств объектов HTML-документа и самого браузера.

Программный код встроенных объектов JavaScript недоступен. Встроенные объекты имеют фиксированные названия. Наиболее важными объектами в разработке web-сайтов являются String (символьные строки), Array (массивы), Math (математические формулы и константы), Date (работа с датами).

Объекты, названия которых совпадают с их фиксированными названиями, называются статическими. Можно создавать и экземпляры (копии) статических объектов, имеющие собственные имена и наследующие все свойства и методы статических объектов.

Встроенные объекты имеют прототипы (prototype), позволяющие добавлять новые свойства и методы к уже существующим – в экземплярах объектов.

Объект String

С помощью объекта String можно создавать строку или строковый объект. Синтаксис:

имя_переменной = new String ("значение")

Создать строковый объект можно и с помощью обычного оператора присваивания:

имя_переменной = "значение"

или

var имя_переменной = "значение"

Например.

mystring1 = new String ("строка")

mystring2 = "строка"

Доступ к свойствам и методам объекта осуществляется через операцию «точка».

Объект String имеет свойство length – длина (количество символов, включая пробелы).

Например:

```
str = "весна"
str.length        // значение равно 5
"лето".length     // значение равно 4
```

Методы String

Как и любые функции, методы могут иметь параметры. Параметры перечисляются через запятую в круглых скобках, стоящих после имени метода.

Метод charAt (индекс) возвращает символ, занимающий в строке указанную позицию. Индекс является числом. Необходимо помнить, что нумерация элементов строки начинается с нуля.

Примеры

```
y = "Осень".charAt(3)        // значение "и"
```

```
str = "Зима"
str.charAt (str . length - 2)      // значение "м"
```

Метод charCodeAt(индекс) преобразует символ в указанной позиции в его код. Поддерживает систему кодов Unicode, NN4 – ISO-Latin1.

Метод fromCharCode(номер [1, номер2[, номер3, …, номерn]]) возвращает строку символов, числовые коды символов которой указаны в строке параметров.

Метод конкатенации (слияние) строк – concat. Его параметром является объект String. Синтаксис применения – строка1.concat(строка2)

Результатом является строка, полученная дописыванием символов строки2 к строке1.

```
x = "Петр"
y = x .concat (" Семенович")        // результат "Петр Семенович"
```

Метод indexOf(строка_поиска [, индекс]) осуществляет поиск строки, указанной в качестве параметра. Метод возвращает индекс первого вхождения строки. Поиск в пустой строке возвращает -1. Второй параметр не является обязательным, о чем свидетельствуют квадратные скобки. Индекс указывает позицию, с которой начинается поиск.

Метод lastIndexOf (строка_поиска [, индекс]) предусматривает поиск первого вхождения строки, указанной параметром. При этом поиск начинается с конца исходной строки, но возвращаемый индекс отсчитывается сначала.

Метод slice(индекс1[, индекс2]) возвращает подстроку исходной строки, начальный и конечный индексы которой указываются параметрами, за исключением последнего символа. Второй параметр не обязателен. Если он не указан – подразумевается до конца строки.

Метод split(разделитель [, ограничитель]) возвращает массив элементов, полученных из исходной строки. Первый параметр – строка символов, используемая в качестве разделителя строки на элементы. Второй параметр – число, указывающее количество элементов возвращаемого массива из строки, полученной при разделении. Второй параметр не обязателен. Если разделитель – пустая строка, возвращается массив символов строки.

Например:

```
x = "Привет всем!"
x.split(" ")        // значение – массив из элементов "Привет", "всем!"
x. split("е")       // значение – массив из элементов "Прив", "т вс", "м!"
```

Метод substr(индекс[, длина]) – возвращает подстроку исходной строки, начальный индекс и длина, которой указываются параметрами. Если второй параметр не указан, возвращается подстрока с начальной позиции до конца.

Метод substring(индекс1, индекс2) возвращает подстроку исходной строки с позиции индекс1 до позиции индекс2.

Методы toLocaleLowerCase( ), toLowerCase( ) переводят строку в нижний регистр.

Методы toLocaleUpperCase( ), toUpperCase ( ) переводят строку в верхний регистр.

Тексты web-страниц, как правило, создаются и форматируются с помощью тегов HTML. Это же можно сделать средствами JavaScript.

Например, для вывода строки полужирным шрифтом используют метод bold( ). Данный метод не выводит строку в окно браузера, а лишь форматирует ее. Для вывода строки в HTML-документе используется метод write( ) объекта document:

```
<HTML>
<SCRIPT>
st = "Доброе утро! ".bold( )
document. write(st)
</SCRIPT>
</HTML>
```

Методы форматирования строк носят названия, соответствующие тегам HTML:

```
anchor("anchor_имя")
blinc( )
```

bold( )
fixed( )
fontcolor(значение цвета)
fontsize(число от 1 до 7)
italics( )
link(расположение или URL)
big( )
small( )
strike( )
sub( )
sup( )

Объект Array

Массив представляет собой нумерованный набор данных. Элементы массива в JavaScript могут быть разного типа. Нумерация элементов массива начинается с нуля. К элементам массива можно обращаться по их порядковому номеру, заключив его в квадратные скобки, расположенные после имени массива.

Существует несколько способов создания массива:

1. Объявление массива вида

имя_массива = new Array ([длина_массива])

Если длина массива не указана, создается пустой массив, не содержащий ни одного элемента. Иначе создается массив указанной длины, все элементы которого имеют значение null. Создав пустой массив, можно присвоить значения его элементам операцией присваивания.

2. Инициализация массива при объявлении:

имя_массива = new Array (значение1[, значение2[, ... значениеn]])

3. Инициализация каждого отдельного элемента массива, подобно свойствам объекта

имя_массива = new Array( )

имя_массива. имя_элемента1 = значение1

[имя_массива. имя_элемента2 = значение2

[... имя_массива. имя_элемента *n* = значение*n*]]

Например:

```
child = new Array (4)
child[0]= "Женя"
child[1]= 22
child[2]= " июнь"
child[3]= 1996
child = new Array ("Женя", 22, "июнь", 1996)
y=child.length     // y=4

child = new Array ( )
child.name = "Женя"
child.day= 22
child.month= "июнь"
child.year= 1996
```

Свойство length возвращает количество элементов объекта Array.

Свойство prototype позволяет добавлять новые свойства и методы для всех созданных массивов.

Например:

```
function SumNegative (massiv)
{var s = 0
 for (i=0; i<=massiv.length-1; i++)
      if (massiv[i]<0 s +=massiv[i]
     return s
}
mass = new Array(1, -9, 7, -23, -3)
Array.prototype.SumN = SumNegative      //добавляем метод к объекту
Massiv.SumN(mass)              // применяем метод SumN к массиву mass
```

Для создания многомерного массива требуется указать все его размерности, заключив каждую из них в квадратные скобки.

```
matrix = new Array ( )
matrix[0] = new Array ( 2, 3, 8)
matrix[1] = new Array ( 1, -3, 7)
matrix[2] = new Array ( 0, 2, -6)       // массив размерности 3 на 3
```

Методы объекта Array

concat( ) объединяет два массива в третий и возвращает полученный массив.

Синтаксис: имя_массива3 = имя_массива1.concat(имя_массива2)

join( ) создает строку из элементов массива с указанным разделителем между ними, возвращает строку символов.

Синтаксис: имя_массива2 = имя_массива1.join(строка)

pop( ) удаляет последний элемент массива и возвращает его значение.

Синтаксис: имя_массива1.pop( )

push( ) добавляет к массиву последний элемент, значение которого указанно в качестве параметра, и возвращает новую длину массива.

shift( ) удаляет первый элемент массива и возвращает его значение.

unshift( ) добавляет к массиву первый элемент, значение которого указано в качестве аргумента.

reserve( ) переписывает массив в обратном порядке, возвращает массив.

slise(индекс1[, индекс2]) – создает массив из элементов исходного массива с индексами указанного диапазона. Возвращает массив. Если второй индекс не указан, то новый массив создается из элементов с индекса1 до конца исходного массива.

sort( ) упорядочивает элементы массива. Если параметр не указан, сортировка производится на основе ASCII-кодов символов значений, что удобно для строк, но не подходит для чисел. Параметром может служить имя функции, сравнивающей два элемента массива.

Пример:

```
massiv = new Array (7, 1, 34, 5, 63)
function cmp (x, y)
{ return x - y  }
massiv.sort(cmp)          //массив будет сортироваться по возрастанию
```

Эта функция дает критерий сортировки.

splice(индекс, количество [, элем1[, элем2[, … элемn]]]) удаляет (заменяет) из массива элементы. Возвращает массив из удаленных элементов. Первый параметр является индексом первого удаляемого элемента, второй – количеством удаляемых элементов. Если указаны необязательные параметры, то происходит замена элементов указанного диапазона на указанные значения параметров. Но это справедливо, если второй параметр не равен нулю.

```
b = new Array( "один", 2, 3,  4, "пять")
c = b.splice(1, 3, "два",  "три", "четыре")
// массив b из элементов «один», «два», «три», «четыре», «пять»
// массив c из элементов 2, 3, 4
```

toLocaleString( ), toString( ) – преобразуют содержимое массива в символьную строку.

Задание

1. Создать пользовательскую функцию для работы со строками с использованием методов объекта String.

Вариант 1. Функция вставки новой строки в исходную строку. Функция должна иметь три параметра: исходную строку, вставляемую строку и позицию вставки.

Вариант 2. Функция замены в исходной строке всех вхождений заданной подстроки на подстроку замены. Функция должна иметь три параметра: исходную

строку, заменяемую подстроку и подстроку замены.

Вариант 3. Функция удаления лишних пробелов в начале исходной строки.

Вариант 4. Функция удаления всех пробелов с заданной позиции строки.

Вариант 5. Функция удаления слов, длина которых меньше заданного размера. Функция должна иметь два параметра: исходную строку и длину удаляемых слов.

Вариант 6. Функция удаления в строке одинаковых слов.

Вариант 7. Функция удвоения в строке каждого символа, заданного параметром функции.

Вариант 8. Функция удаления симметричных слов в исходной строке.

Вариант 9. Функция удаления слов, длина которых больше заданного размера. Функция должна иметь два параметра: исходную строку и длину удаляемых слов.

Вариант 10. Функция дополнения строки до определенной длины. Параметрами функции определить требуемую длину строки и символ, используемый для дополнения.

2. Изучить функции форматирования строк. Продемонстрировать работу функции из задания 1, отформатировав вновь полученную строку: переведя ее из нижнего регистра в верхний и обратно, изменив ее цвет и используя различные форматы. Использовать для этого как средства HTML, так и JavaScript.

3. Создать функцию для работы с объектом Array.

Вариант 1. Замена минимального элемента значением, заданным как параметр функции, и сортировка элементов массива, расположенных после минимального, по возрастанию.

Вариант 2. Сортировка по возрастанию элементов массива, расположенных между максимальным и минимальным его элементами.

Вариант 3. Сортировка по убыванию элементов массива, расположенных до максимального значения.

Вариант 4. Удаление из массива минимального элемента и добавление утроенного найденного значения в качестве первого элемента массива.

Вариант 5. Удаление максимального элемента массива и добавление найденного значения, умноженного на пять, в качестве последнего элемента массива.

Вариант 6. Создание массива из элементов исходного, расположенных между максимальным и минимальным его элементами.

Вариант 7. Создание массива из элементов исходного, расположенных между первым и пятым отрицательными элементами.

Вариант 8. Сортировка по возрастанию элементов массива, начиная с индекса, заданного в качестве параметра.

Вариант 9. Сортировка по убыванию всех элементов, кроме последнего и предпоследнего.

Вариант 10. Замена последнего элемента первым, предпоследнего – вторым и т.д.

4. Добавить созданную функцию из задания 3 в качестве нового свойства ко всем объектам Array.

Контрольные вопросы

1. Поясните понятия: класс, объект, свойство, метод.
2. Перечислите особенности встроенных объектов.
3. Поясните действие методов split( ), substring( ), astIndex0( ). Перечислите их параметры.
4. Каковы способы создания объектов типа Array?
5. Поясните действие методов pop( ), push( ), shift( ), reserve( ), slise( ). Запишите синтаксис их вызова.
6. Возможно ли создание массива из элементов различных типов? Приведите примеры.

Лабораторная работа № 4

Обработка событий. Простые визуальные эффекты

Программы, работающие с объектами HTML-документа, называют сценариями. Стандартным является размещение сценария в контейнерном теге <SCRIPT> </SCRIPT>. Этот контейнер может располагаться в любом месте HTML-документа, причем любое количество раз. От его расположения иногда может зависеть функционирование всего HTML-документа. Атрибутами контейнера <SCRIPT> являются:

1. LANGUAGE – язык сценария, его возможные значения

"JavaScript", "JScript";

"VBScript", "VBS".

Если атрибут LANGUAGE не указан, подразумевается JScript.

2. SRC – указывает файл (имя или URL-адрес), содержащий код сценария. Этот атрибут используется только если сценарий расположен не в HTML-документе. Если сценарий располагается в отдельном файле, то его вызов выглядит следующим образом:

<SCRIPT SRC = "file1.js"></SCRIPT>

Одним из главных назначений сценариев в HTML-документе является обработка событий – таких как щелчок кнопки мыши по элементу документа, помещение указателя мыши на элемент, нажатие клавиши и др. Для одного и того же элемента можно определить несколько событий, на которые он будет реагировать. Список всех допустимых событий довольно обширен. События обозначаются соответствующими словами на английском языке. Например, щелчок левой кнопкой мыши – onclick, изменение в поле ввода данных – onchange, помещение указателя мыши на элемент – onmouseover.

Для одного и того же элемента можно определить ряд событий, на которые он будет реагировать.

Значением события является код сценария, заключенный в кавычки. Обыч-

но обработчики событий оформляются в виде функций, определение которых помещают в контейнер <SCRIPT>.

В качестве примера рассмотрим оформление обработчика щелчка мыши по изображению. Изображение в HTML-документе определяется тегом <IMG>. Файл с изображением задается атрибутом SRC. Обработчик события onclick зададим функцией clickimage(), которая должна быть определена в контейнере <SCRIPT>. В результате щелчка на графическом изображении из файла p.jpg выводится окно с сообщением.

```
<HTML>
<SCRIPT>
function clickimage( ) {
alert("Чего изволите?")
}
</SCRIPT>
<IMG SRC = "p.jpg" onclick = "clickimage()">
</HTML>
```

Это же можно выполнить иначе:

```
<HTML>
<IMG SRC = "pict.jpg" onclick = "alert("Чего изволите") ">
</HTML>
```

Такой способ удобен, если сценарий обработки события небольшой и используется в HTML-документе один раз. В этом случае событие является атрибутом. В других случаях предпочтительнее оформление обработчика в виде функции.

Можно явным образом указать браузеру, что сценарий написан на языке JavaScript, для этого в значении атрибута-события написать префикс "javascript:"

```
<IMG onclick = "javascript: alert("Чего изволите?")">
```

Еще один способ оформления обработчиков событий заключается в использовании атрибута - ID (идентификатор). Этот атрибут принимает любые строковые значения, которые играют роль индивидуальных имен элементов в их объ-

ектном представлении. При использовании атрибута ID в теге в задании обработчика события можно не использовать атрибуты-события. Вместо этого в контейнере <SCRIPT> достаточно написать определение функции обработчика события:

```
<HTML>
<H1  ID =  "Myheader">Привет  всем!</H1>
<SCRIPT>
function Myheader.onclick( ) {
alert("Привет!")
}
</SCRIPT>
</HTML>
```

Требуется помнить, что элемент HTML-документа должен быть загружен раньше, чем функция-обработчик события, а составное имя функции-обработчика события содержит название события в нижнем регистре.

К простым визуальным эффектам относят смену изображений по некоторому событию, изменение цвета шрифта, ссылок, кнопок, использование объемных заголовков и пр. Некоторые из них применяются довольно часто, другие редко. Все они могут быть реализованы сценариями.

Для смены одного изображения на другое достаточно с помощью сценария заменить значение атрибута SCR тега <IMG>. Например:

```
<HTML>
<IMG ID = "myimg" SRC= 'pict1.gif '
onclick = "document.all.myimg.src = 'pict2.gif' ">
</HTML>
```

Замена изображения из файла pict1.gif на другое из файла pict2 происходит при первом щелчке мыши на нем. Последующие щелчки не приведут к видимым изменениям, поскольку второе изображение будет заменяться им же. Чтобы при повторном щелчке происходила замена изображения на предыдущее, необходимо создать переменную-триггер (флаг), принимающий одно из двух возможных зна-

чений, по которому можно определить, какое из двух значений надо отобразить.

```
<HTML>
<IMG ID = "myimg" SRC= 'pict1.gif '
onclick = " imgchange( ) ">
<SCRIPT>
var flag=false
function  imgchange( ) {
if (flag) document. all. myimg. src = "pict1.gif"
      else document. all. myimg. src = "pict2.gif"
flag=!flag
}
</SCRIPT>
</HTML>
```

Задача изменения цвета кнопки при наведении на нее указателя мыши и возвращения в первоначальное состояние при удалении указателя с кнопки может быть решена следующим образом:

```
<HTML>
<STYLE>
mystile {font-weight.bold; background-color: a0a0ao} // серый цвет кнопок
</STYLE>
<FORM onmouseover = "colorchange ( 'yellow') " onmouseout
= "colorchange ('a0a0a0') ">
<INPUT TYPE = "BUTTON" VALUE = "Кнопка" CLASS = "mystile"
onclick = "alert( 'Вы нажали кнопку') " >
</FORM>
<SCRIPT>
function colorchange (color){
if (event. srcElement.type = = "button")
event. srcElement. style. backgroundColor = color;
```

```
}
</SCRIPT>
</HTML>
```

Функция colorchange( ) проверяет, является ли инициатор события объектом типа button. Если это так, то цвет кнопки меняется. Без этой проверки менялся бы не только цвет кнопок, но и текста.

Аналогичным способом можно изменять цвет фрагментов текста. Но в этом случае текст должен быть заключен в контейнер, – например, в теги <P>, <B>, <I>, <DIV>.

Можно создать прямоугольную рамку, окаймляющую текст, которая периодически изменяет цвет. Рамка создается тегами одноячеечной таблицы с заданием нужных атрибутов и параметров стиля:

```
<TABLE ID= "tab" BORDER=1 WIDH=200 style= "border:10 solid : red">
<TR><TD> Доброе утро! </TR></TD>
</TABLE>
```

Функция изменения цвета:

```
<SCRIPT>
function flash( ) {
if ( !document.all) return null;
if (tab.style.borderColor = = 'red')    tab.style.borderColor = 'yellow'
else tab.style.borderColor = 'red';
}
setInterval ("flash()", 500);                //мигание рамки с интервалом 500 мс
</SCRIPT>
```

Для изменения цвета случайным образом можно использовать метод random( ) (счетчик случайных чисел) встроенного объекта Math. Если требуется получить случайное число x, лежащее в интервале от A до B, то x=A+(B – A)*Math.random( ).

Объемные заголовки формируются наложением нескольких надписей с

одинаковым содержанием с некоторым сдвигом по координатам. Наилучший эффект достигается путем подбора цветов надписей (игрой света и тени) с учетом цвета фона. Для этого используют библиотеку стилей. Функция, создающая заголовок с заданными параметрами:

```
function d3 (text, x, y, tcolor, fsize, fweight, family, zind) {
/*   text – текс заголовка
     x – горизонтальная координата (left)
     y – вертикальная координата (top)
     tcolor – цвет переднего плана
     fsize – размер шрифта (пт)
     fweight – вес (толщина шрифта)
     family – название семейства шрифтов
     zind z-Index                */
if (!text) return null        // если текст не указан, ничего не выполняется
//значения параметров по умолчанию
if (!x) x=0
if (!y) y=0
if (!tcolor) tcolor='00aaff'
if (!fsize)  fsize=36
if (!fweight)  fweight =800
if (!family) ffamily='arial'
// внутренние настройки
var sd=5, bd=2      // сдвиг тени и подсветки
var xzind=" "
if (zind) xzind= ';  z-Index: ' +zind
var xstyle ='font-family:' + family + ';font-size:' + fsize +
      ';font-weight:' + fweight + ';'
var xstr = '<DIV STYLE = "position: absolute; top:'  + (y -sd ) +
      '; left :' + ( x + sd ) +  xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color: darked">' + text + '</P></DIV>'
```

```
xstr+= '<DIV STYLE = " position: absolute; top:' + y  + '; left :' +
x  + xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color: silver">' + text + '</P></DIV>'
xstr+= '<DIV STYLE = " position: absolute; top :' + ( y + hd )  + '; left :' +
(x +hd)  +  xzind + ' ">'
xstr+='<P styl e  = " ' + xstyle + 'color:' + tcolor +  '">' + text + '</P></DIV>'
document.write(xstr)            //запись в документ
}
```

Параметр z-Index позволяет установить слой, в котором находится заголовок, и тем самым указать, будет ли заголовок располагаться над или под другим видимым элементом документа. Элементы с более высоким значением z-Index находятся над элементами, у которых z-Index меньше. Перекрытие элементов с одинаковыми значениями z-Index определяется порядком их следования в HTML-документе.

Вызов приведенной выше функции может выглядеть так:

```
d3 ("это объемный заголовок", 50, 50, 'blue', 72, 800, 'times')
```

Задание

С использованием сценария создать программу для работы с галереей миниатюр. При щелчке кнопкой мыши по миниатюре изображение должно увеличиваться, а затем при щелчке на увеличенном изображении – уменьшаться. Доработать приведенную в тексте функцию imgchange( ). Для решения этой задачи использовать массив флагов и функцию обработчик, определяющую на изображений произошел щелчок:

```
var p1 =new Array ("p1.gif" , ... ) //массив имен исходных файлов
var p2=new Array ("p2.gif" , ... ) //массив имен замещающих файлов
//формирование тегов, описывающих изображения
var xstr = " "
for (i=0; i<p1.length; i++)
xstr+= '<IMG ID = "i' + i +' " SRC = " '+ p1[i]+' " onclick = "imgchange( )" >'
```



```
}
document.write(xstr)            //запись в документ
```

Вариант 1. Выполнить замену цвета заголовка галереи при наведении на него указателя мыши с синего на красный. Сопроводить изображения текстом. Выделить фрагмент текста мигающей трехцветной рамкой.

Вариант 2. Поместить под изображениями текст со ссылками. Создать эффект динамического изменения цвета ссылок. Различать цвета мерцания использованных и неиспользованных ссылок. (Множество цветов задать массивом. Использовать свойства linkColor объекта document).

Вариант 3. В тексте документа создать три объемных заголовка разных уровней, поэкспериментировав со значениями внутренних параметров sd, hd, а также с заданием параметров по умолчанию. Изменить цвет заголовка верхнего уровня при наведении на него курсора мыши.

Вариант 4. Сопроводить изображения текстом, расположив его над изображениями и выделив мигающей трехцветной рамкой. Заголовок текста сделать объемным.

Вариант 5. Поместить над изображениями текст со ссылками. Создать эффект динамического изменения цвета ссылок. Различать цвета мерцания использованных и неиспользованных ссылок. Множество цветов задать массивом. Использовать свойства linkColor объекта document.

Вариант 6. Сопроводить изображения текстом с объемным заголовком. Создать две кнопки. Первая изменяет цвет текста, вторая меняет фон текстовой области. При использовании кнопок изменить их цвет.

Вариант 7. Выполнить замену цвета заголовка галереи при наведении на него указателя мыши и возвращение его в исходный цвет при повторном наведении. Создать две кнопки. Первая убирает галерею, вторая возвращает. Менять цвет кнопок при их использовании.

Вариант 8. Создать два объемных заголовка, поэкспериментировав со значениями внутренних параметров sd, hd, а также с заданием параметров по умолчанию. Выделить фрагмент текста мигающей трехцветной рамкой, исчезающей

при наведении на него курсором мыши.

Вариант 9. Поместить сбоку от изображений текст со ссылками. Создать эффект динамического изменения цвета ссылок. Различать цвета мерцания использованных и неиспользованных ссылок. Множество цветов задать счетчиком случайным чисел. Использовать свойства linkColor объекта document.

Вариант 10. В тексте документа создать три объемных заголовка разных уровней, поэкспериментировав со значениями внутренних параметров sd, hd, а также с заданием параметров по умолчанию. Создать кнопки, при нажатии которых заголовки меняют цвет на другой, заданный случайным образом.

Контрольные вопросы

1. Перечислите атрибутами контейнера <SCRIPT>
2. Поясните механизм формирования объемного заголовка.
3. Приведите примеры простых визуальных эффектов.

Лабораторная работа №5

Работа с окнами. Объектная модель документа

Главное окно браузера создается автоматически при запуске браузера. С помощью сценария можно создавать любое количество окон, а также разбивать окно на несколько прямоугольных областей, называемых фреймами. Окну браузера соответствует объект window, а HTML-документу, загруженному в окно, соответствует объект document. Эти объекты могут содержать в себе другие объекты, в частности, объект document входит в состав window.

Доступ к свойствам и методам объекта происходит через точку. Поскольку объект document является подобъектом объекта window, ссылка на HTML-документ, загруженный в текущее окно: window.document.

Объект окна window – корневой объект, имеющий свои подобъекты. Например, location хранит информацию об URL-адресе загруженного документа,

screen – данные о возможностях экрана монитора пользователя.

Все множество объектов имеет иерархическую структуру, называемую объектной моделью. Объекты могут находиться в отношении вложенности. Если объект входит в состав другого, его иногда называют подобъектом, или свойством.

В объектной модели документа объекты группируются в коллекции. Коллекция – промежуточный объект, содержащий объекты документа, отсортированные в порядке упоминания соответствующих им элементов в HTML-документе. Индексация объектов в коллекции начинается с нуля. Синтаксис обращения к элементам коллекции аналогичен синтаксису обращению к элементам массива. Коллекция имеет свойство length – длину. Коллекция всех графических изображений документа называется images, коллекция всех форм – forms, ссылок – links. Коллекция всех объектов документа называется all.

Основные элементы объектной модели браузера можно представить в виде, представленном на рис. 4.

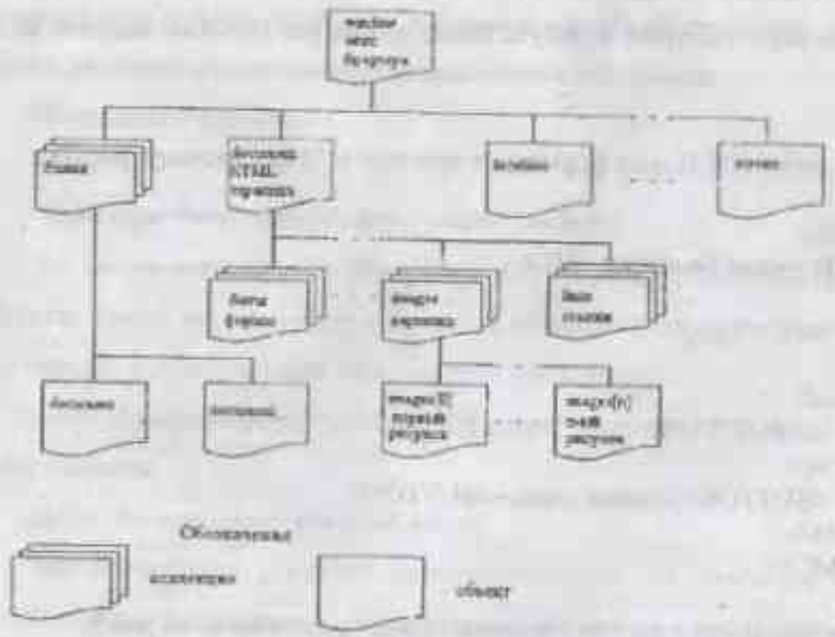


Рис. 4. Примерная объекта модель документа

Один и тот же объект может входить в частную коллекцию (например, images), но он обязательно входит в коллекцию all. При этом его индексы могут быть разными в разных коллекциях.

При использовании документа, загруженного в текущее окно, объект window можно не упоминать, а сразу начинать с объекта document.

Например:

document. images[0]

Существует несколько способов для обращения к объекту документа:

document. коллекция.id_объекта;

document. коллекция [id_объекта];

document. коллекция.[индекс_объекта].

Здесь id_объекта является значением атрибута ID в теге, определяющем соответствующий элемент HTML-документа. Если при создании HTML-документа атрибут ID для некоторых элементов не использовался, то обращение к ним возможно только через индексы. Такие теги как <FORM> и <INPUT> имеют атрибут NAME, тогда значение этого атрибута также можно использовать наравне со значением ID.

Следующий HTML-код формирует простую HTML-страницу (рис.5).

```
<HTML>
<H1> Изучаем JavaScript <H1>
<p>
<IMG SRC="2.jpg">
<p>
<FORM>
      <INPUT TYPE="text" VALUE="">
      <p>
      <BUTTON> Нажми здесь </BUTTON>
</FORM>
</HTML>
```

Объектная модель данной страницы также представлена на рис 5.

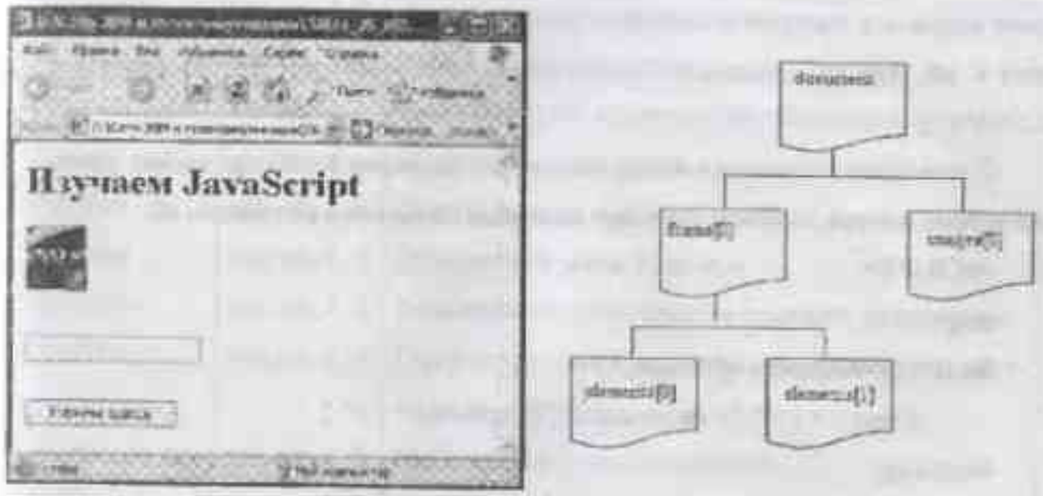


Рис. 5. HTML-страница и соответствующая объектная модель

Если количество страниц велико, то процедура адресации к различным объектам по номеру может стать весьма запутанной. Например, document.forms[7].elements[25]. Во избежание подобной проблемы можно присваивать различным объектам уникальные имена. Например:

```
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
```

Эта запись означает, что объект forms[0] получает теперь еще и второе имя – myForm. Точно так же вместо elements[0] можно использовать name (последнее было указано в атрибуте name тега <input>). Тогда вместо

name= document.forms[0].elements[0].value;

можно записать

name= document.myForm.name.value;

Это значительно упрощает программирование на JavaScript, особенно в случае работы с большими web-страницами, содержащими множество объектов.

Существует универсальный способ обращения к объектам документа – обращение посредством коллекции all. Нужно помнить, что один и тот же объект

может входить в частную коллекцию (например, images), но обязательно содержится в all. При этом индексы объекта в этих коллекциях могут быть различными.

С помощью добавления нижеследующего сценария JavaScript можно узнать порядковые номера, соответствующие индексам объектов в коллекции all.

```
<SCRIPT>
msg=" "
for (i=0; i<document.all.length; i++)
    { msg += i + " "+ document.all[i].tagName + "\n" }
alert(msg)
</SCRIPT>
```

В результате выполнения данного кода в диалоговом окне alert() будут перечислены все теги HTML-документа с порядковыми номерами, соответствующими индексам в коллекции all. Объекты, соответствующие базовым тегам, – таким как <HEAD>, <TITLE>, <BODY>, присутствуют в коллекции all даже в случае отсутствия их явного применения.

С помощью сценария можно создавать любое количество окон. Для этого применяется метод open( ):

window. open(параметры)

Данному методу передаются следующие необязательные параметры:

адрес документа, который нужно загрузить в создаваемое окно;

имя окна (как имя переменной);

строка описания свойств окна (features).

В строке свойств записываются пары свойство = значение, которые отделяются друг от друга запятыми.

Свойства, передаваемые в строке features

Свойство	Значения	Описание
channel mode	yes, no, 1, 0	Показывает элементы управления channel
directories	yes, no, 1, 0	Включают кнопки каталога

fullscreen	yes, no, 1, 0	Полностью разворачивает окно
height	число	Высота окна в пикселях
left	число	Положение по горизонтали относительно левого края экрана в пикселях
location	yes, no, 1, 0	Текстовое поле Address
menubar	yes, no, 1, 0	Стандартное меню браузера
resizeable	yes, no, 1, 0	Возможность пользователя изменять размер окна
scrollbars	yes, no, 1, 0	Горизонтальные и вертикальные полосы прокрутки
status	yes, no, 1, 0	Стандартная строка состояния
toolbar	yes, no, 1, 0	Включает панели инструментов браузера
top	число	Положение по вертикали относительно верхнего края экрана в пикселях
width	число	Ширина окна в пикселях

Примеры

```
window. open ("mypage.htm", "NewWin", "height=150, width=300")
window. open ("mypage.htm" )
strfeatures = "top=100, left=15, height=250, width=300, location=no"
window. open ("www.amsu.ru", strfeatures)
```

Вместо строки strfeatures можно использовать значение true, тогда указанный документ загружается в существующее окно, вытесняя предыдущий документ.

Метод window.open( ) возвращает ссылку на объект окна: если ее сохранить, можно использовать позднее, например, при закрытии окна.

Для закрытия используют метод close( ). Однако выражение window.close( ) закрывает главное окно. Для закрытия других окон используют ссылки.

```
var str = window. open ("mypage.htm", "моя страница")
str.close( )
```

Объект window, кроме дочерних объектов, имеет свои методы, свойства, события. Они указаны в *приложении В* данного пособия.

Например, window.status = "работает сценарий".

Свойство parent позволяет обратиться к объекту, расположенному в иерархии на одну ступень выше. Для перемещения на две ступени выше используют parent.parent. Для обращения к самому главному окну – окну браузера, используют свойство top.

Свойство status применяют для вывода сообщений во время работы сценария. Например, window. status = "сценарий работает".

Рассмотренные выше методы позволяют работать с независимыми (немодальными) окнами. Для создания модального окна используется метод showModalDialog(). В качестве параметра данный метод принимает адрес документа (файла), имя окна и строку свойств.

Свойства модального окна

Свойство	Значения	Описание
border	thick, thin	Размер рамки вокруг окна (толстая/тонкая)
center	yes, no, 1, 0	Выравнивание окна по центру главного
dialogHeight	число + единицы измерения	Высота окна
dialogLeft	число + единицы измерения	Горизонтальная координата
dialogTop	число – единицы измерения	Вертикальная координата
dialogWidth	число + единицы измерения	Ширина окна
font	строка таблицы стилей	Стиль окна, определенный по умолчанию
font-family	строка таблицы стилей	Вид шрифта, определенный по умолчанию для окна
font-size	строка таблицы стилей	Размер шрифта, определенный по

		умолчанию для окна
font-style	строка таблицы стилей	Тип шрифта, определенный по умолчанию для окна
font-variant	строка таблицы стилей	Вариант шрифта, определенный по умолчанию для окна
font-weight	строка таблицы стилей	Толщина шрифта, определенная по умолчанию для окна
help	yes, no, 1, 0	Включение кнопки Help в верхнюю панель
maximize	yes, no, 1, 0	Включение кнопки Maximaze в верхнюю панель
minimize	yes, no, 1, 0	Включение кнопки Minimaze в верхнюю панель

При работе с модальными окнами пользователь не может обратиться к другим окнам, в том числе и к главному. Окна, создаваемые методами alert( ), prompt(), confirm( ), являются модальными.

В случае открытия нескольких окон браузера пользователь может переключаться между ними, переводя фокус с одного окна на другое. Эти действия инициируются программными событиями onblur и onfocus. Эти же действия можно вызвать, используя методы blur и focus.

Событие onerror происходит при ошибке загрузки страницы или ее элемента. Его можно использовать в программе при попытке вновь загрузить страницу.

```
<SCRIPT>
function window.onerror( ) {
alert (" Ошибка! Повтори попытку!")
}
</SCRIPT>
```

Объект document является центральным в иерархической объектной модели.

Он предоставляет всю информацию о HTML-документе с помощью коллекций и свойств и множество методов для работы с документами.

all	все теги и элементы основной части документа
anchor	якоря (закладки) документа
applets	все объекты документа, включая встроенные элементы управления, графические элементы, аплеты, внедренные объекты
embeds	все внедренные объекты документа
forms	все формы на странице
frames	фреймы, определенные в теге <FRAMESET>
images	графические элементы
links	ссылки и блоки <AREA>
plugins	другое название внедренных документов
scripts	все разделы <SCRIPT> на странице
styleSheets	контейнерные свойства стиля, определенные в документе

Свойства, методы и события объекта document представлены в *приложении Г* данного пособия.

Элементы HTML-документа задаются тегами, большинство из которых имеют параметры (атрибуты). В объектной модели документа тегам соответствуют объекты, а атрибутам – свойства этих объектов. Названия свойств объектов, как правило, совпадают с названиями атрибутов, но записываются в нижнем регистре.

Наиболее удобный способ динамического изменения HTML-документа основан на использовании свойств innerText, outerText, innerHTML и outerHTML. С их помощью можно получить доступ к содержимому элемента. Изменяя значения перечисленных свойств, можно частично или полностью изменить сам элемент. Например, можно изменить только надпись на кнопке, а можно превратить кнопку в изображение, или Flash-анимацию.

Значением свойства innerText является все текстовое содержимое между открывающим и закрывающим тегами элемента. Внутренние теги игнорируются.

Данные открывающего и закрывающего тегов соответствующего элемента также не входят.

В отличие от предыдущего свойство outerText включает в себя данные открывающего и закрывающего тегов. Таким образом, outerTextесть – весь текст, содержащийся в контейнере, включая его внешние теги. Например:

```
<DIV ID = "my" >
<A HREF = 'raznoe.htm'>
<IMG SRC = 'picture.jpg> Ссылка на раздел <B> Разное <B>
</A>
<DIV>
```

Здесь свойства innerText и outerText для элемента, заданного контейнерным тегом <DIV>, совпадают:

```
document.all.my.innerText    //значение равно – «Ссылка на раздел Разное»
```

При присвоении свойствам innerText и outerText новых значений нужно помнить, что если значения содержат теги, то они не интерпретируются, а воспринимаются как обычный текст.

Свойство innerHTML содержит внутренний HTML-код контейнера элемента. Присвоение этому свойству нового значения, содержащего HTML-код, приводит к интерпретации кода. Свойство outerText дополнительно включает внешние открывающие и закрывающие теги элемента.

Для приведенного HTML-кода значение document. all. my. innerHTML равно " <A HREF = 'raznoe.htm'> <IMG SRC = 'picture.jpg'> Ссылка на раздел <B> Разное <\B> </A>"/

Значение document. all. my. outerHTML – "<DIV ID = "my" > <A HREF = 'raznoe.htm'> <IMG SRC = 'picture.jpg> Ссылка на раздел <B> Разное <\B> </A> <DIV>".

Если в сценарии выполнить выражение

```
document.all.my.innerHTML = "<BUTTON>Щелкни здесь</BUTTON>"
```

ссылка, изображение и текст будут заменены кнопкой с надписью «Щелкни здесь». При этом контейнерный тег "<DIV ID = "my" > сохранится. Если анало-

гичным образом использовать outerHTML, кнопка также появится, но уже без контейнера "<DIV ID = "my" >.

Свойства innerHTML и outerHTML могут применяться к элементам, заданным неконтейнерными тегами. Тогда innerHTML и outerHTML совпадают.

Для ускорения загрузки графики можно использовать следующие возможности JavaScript. Можно организовать предварительную загрузку изображений в кэш-память браузера, не отображая их на экране. Это особенно эффективно при начальной загрузке страницы. Пока изображения загружаются в память, оставаясь невидимыми, пользователь может рассматривать текстовую информацию.

Для предварительной загрузки изображения требуется создать его объект в памяти браузера. Это можно сделать следующим выражением:

myImg = new Image (ширина, высота)

Параметры должны соответствовать значениям атрибутов WIDTH и HEIGHT тега <IMG>, который используется для отображения предварительно загруженного изображения.

Для созданного в памяти объекта изображения можно создать имя или URL-адрес графического файла:

myImg.scr = "URL-адрес изображения"

что предписывает браузеру загрузить изображения без его отображения.

После загрузки в кэш-память всех изображений и загрузки всего документа можно сделать их видимыми. Для этого свойству src элемента <IMG> нужно присвоить значение этого же свойства объекта изображения в кэш-памяти. Например:

document. images[0]. scr = myImg. scr

Здесь слева указано свойство scr первого в документе элемента, соответствующего тега <IMG>, справа – свойство scr объекта изображения в кэш-памяти.

С помощью JavaScript можно через заданный интервал времени запускать код или функцию. При этом создается эффект одновременного (параллельного) выполнения вычислительных процессов.

Для организации повторения через заданный интервал выполнения некото-

рого выражения служит метод setInterval( ) объекта window:

setInterval(выражение, период, [, язык])

Первым параметром является строка, – например вызов функции. Период указывается в миллисекундах. Третий параметр – необязательный, в нем указывается язык, с помощью которого написано заданное выражение. По умолчанию – JavaScript.

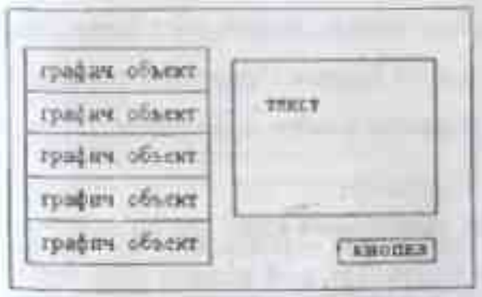
Метод setInterval( ) возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем, – например для прекращения выполнения процесса методом clearInterval( ):

```
var pr = setInterval( "myfunc(), 100" )
if (confirm ( "Прервать процесс?") )
clearInterval(pr)
```

Если требуется выполнить действие с некоторой временной задержкой, применяется метод setTimeout( ), имеющий синтаксис, аналогичный setInterval( ). Для отмены задержки процесса, запущенного setTimeout( ), используют clearTimeout( ).

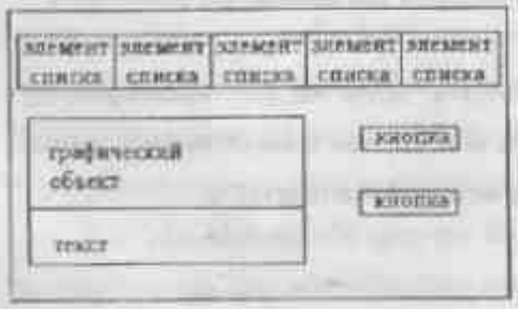
Задание

Вариант 1. Создать HTML-документ, расположив в нем вертикально пять графических объектов, одно текстовое окно, кнопку. Щелчок по выбранному графическому объекту должен приводить к замене текста. Содержание текста должно давать характеристику изображенного объекта. Внутри текста должны быть предусмотрены ссылки, меняющие свой цвет при использовании. Щелчок по кнопке должен инициализировать функцию открытия дополнительного документа в окне, заданного размера. Окно должно содержать горизонтальные полосы прокрутки. В окне должны отображаться

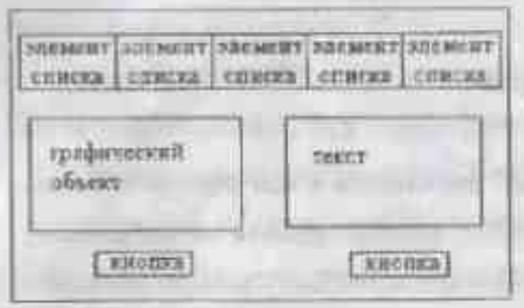


последовательно все текстовые сопровождения графических объектов. Продемонстрировать дополнительно по три события и свойства объектов window и document по выбору. Составить объектную модель документа.

Вариант 2. Создать HTML-документ, расположив в нем горизонтальный список пяти названий графических объектов, одно исходное отображение, две кнопки. Щелчок на элементе списка должен приводить к отображению соответствующего графического объекта и текстового сопровождения. По нажатию первой кнопки изображение должно закрываться, по нажатию второй – появляться. Продемонстрировать по дополнительно три события и свойства объектов window и document. Составить объектную модель документа.



Вариант 3. Создать HTML-документ, расположив в нем горизонтальный список названий графических объектов, одно исходное отображение, две кнопки. По нажатию кнопки через три секунды изображения графических объектов должны циклически сменять друг друга. При выборе элемента из списка должно появляться соответствующее текстовое сопровождение. Нажатие кнопки должно менять цвет фона. Это действие может быть отменено с помощью второй кнопки. Продемонстрировать дополнительно по три события и свойства объектов window и document. Составить объектную модель документа.



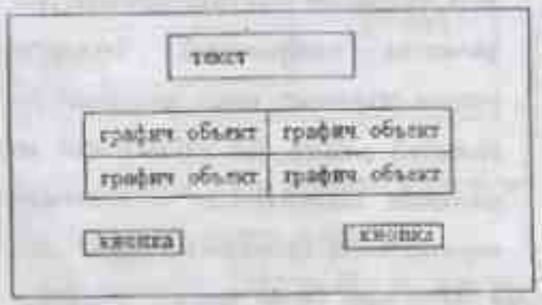
Вариант 4. Создать HTML-документ, расположив в нем три кнопки. Нажатие первой приводит к появлению в окне текста, который должен быть красным на сером фоне, иметь выделенный заголовок, ссылки на графические объекты. Окно текста должно иметь заданные размеры и полосы вертикальной и горизонтальной прокрутки. При нажатии второй кнопки цвета кнопки и текста должны меняться. При нажатии третьей кнопки через три секунды должно появляться графическое изображение вместо текста. Продемонстрировать по три события и свойства объектов window и document по выбору. Составить объектную модель документа.

Вариант 5. Создать HTML-документ, расположив в нем пять кнопок по горизонтали. Ниже расположить три графических объекта. По нажатию любой из первых трех кнопок должны происходить изменения соответствующих графических изображений. Четвертая кнопка открывает текст, имеющий небольшой размер, не изменяемый по желанию пользователя, с полосами вертикальной прокрутки. Текст должен иметь выделенные разными цветами заголовки. Пятая кнопка закрывает текст. Продемонстрировать по три события и свойства объектов window и document по выбору. Составить объектную модель документа.

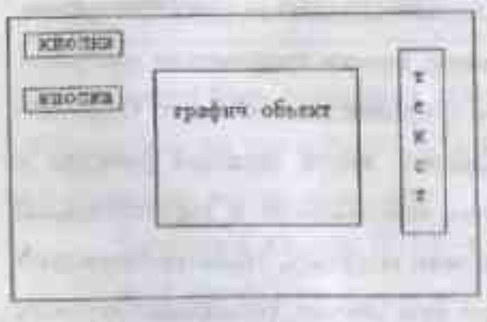
Вариант 6. Создать HTML-документ, расположив в нем пять кнопок по вертикали. Напротив кнопок расположить область отображения графического объекта, а под ней область текста. При нажатии любой из первых четырех кнопок должно появляться соответствующее названию кнопки графическое изображение и сопровождающий текст. Окно текста должно иметь заданные размеры и полосы вертикальной и горизонтальной прокрутки. При нажатии кнопки ее цвет должен меняться. Нажатие последней кнопки закрывает весь HTML-документ через пять секунд. Продемонстрировать по три события и свойства объектов window и document по выбору. Составить объектную модель документа.

Вариант 7. Создать HTML-документ, расположив в его окне текст, содержащий не менее трех ссылок и трех графических объектов. При наведении мыши на ссылку ее цвет меняется. По ссылке происходит открытие документов, закрытие – через пять секунд, после нажатия специально выделенной кнопки. Использованные ссылки должны быть выделены другим цветом. Продемонстрировать по три события и свойства объектов window и document по выбору. Составить объектную модель документа.

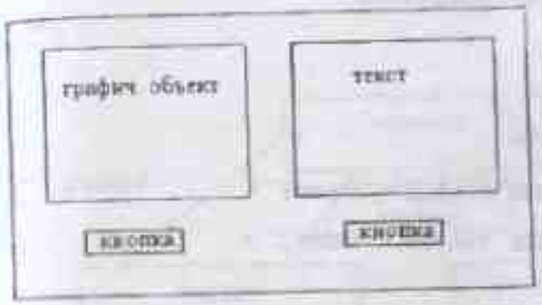
Вариант 8. Создать HTML-документ, расположив в его центре четыре графических объекта, текстовое окно и две кнопки. Текстовое окно имеет небольшой размер, не изменяемый по желанию пользователя, с полосами вертикальной прокрутки. Щелчок по графическому объекту должен инициировать открытие дополнительного окна, содержащего увеличенный графический объект. Первая кнопка закрывает и вновь проявляет графические объекты, вторая – открывает и закрывает окно текста. Продемонстрировать дополнительно по три события и свойства объектов window и document. Составить объектную модель документа.



Вариант 9. Создать HTML-документ, расположив в нем две кнопки, окно графического объекта, окно текста. Текст должен иметь выделенный цветом заголовок. Окно текста изменяется по желанию пользователя, имеет полосы горизонтальной прокрутки. В окне графического объекта обеспечить циклические повторения трех картинок. Первая из кнопок останавливает смену картинок. Вторая – возобновляет. Продемонстрировать дополнительно по три события и свойства объектов window и document. Составить объектную модель документа.



Вариант 10. Создать HTML-документ, расположив в нем окно текста, окно размещения графического объекта, две кнопки. Текст должен иметь выделенные заголовки и пять ссылок. Окно текста должно иметь неизменяемый пользователем размер, полосы вертикальной и горизонтальной прокрутки. Ссылки должны менять цвет после их использования. По нажатию ссылки в окне появляется соответствующее изображение. Одна из кнопок должна менять фон текстовой области, другая – цвет текста. Продемонстрировать дополнительно по три события и свойства объектов window и document. Составить объектную модель документа.



Контрольные вопросы

1. Назовите корневой объект HTML-документа.
2. Каким образом оформляются обработчики событий?
3. Какими механизмами можно воспользоваться для ускорения загрузки графики?
4. Перечислите методы объекта window.
5. Какие методы используются для задержки во времени событий?

Лабораторная работа №6

Работа с фреймами

Фрейм – прямоугольная область окна браузера, в которую можно загрузить HTML-документ. В общем случае окно браузера может быть разбито на несколько отдельных фреймов. Это означает, что фрейм определяется как некое выделенное в окне браузера поле в форме прямоугольника. Каждый из фреймов выдает на экран содержимое собственного документа (в большинстве случаев это документы HTML). Разбиение окна браузера на отдельные окна производится с помощью

тега <FRAMESET>, внутрь которого вставляются теги <FRAME> с атрибутами, указывающими имя фрейма и адрес HTML-документа.

Пример

```
<HTML>
<FRAMESET ROWS= "30%, 70%">
<FRAMESET SRC= "документ1.htm" NAME = "frame1" >
<FRAMESET SRC= "документ2.htm" NAME = "frame2" >
</FRAMESET>
</HTML>
```

Здесь применяется вертикальное расположение фреймов. Для горизонтального размещения фреймов вместо атрибута ROWS в теге <FRAMESET> использовать COLS.

Используя вложение тега <FRAMESET>, можно разбить уже имеющийся фрейм на другие.

```
<HTML>
<FRAMESET COLS= "50%, 50%">
<FRAMESET ROWS= "50%, 50%">
<FRAMESET SRC= "документ1.htm" >
<FRAMESET SRC= "документ1.htm" >
</FRAMESET>
<FRAMESET ROWS= "25%, 25%, 25%, 25%">
<FRAMESET SRC= "документ1.htm" >
<FRAMESET SRC= "документ1.htm" >
<FRAMESET SRC= "документ1.htm" >
<FRAMESET SRC= "документ1.htm" >
</FRAMESET>
</FRAMESET>
</HTML>
```

При разбиении окна на фреймы и в свою очередь фрейма на другие фреймы возникают отношения родитель – потомок. Каждому из фреймов соответствует

свой объект document. Обеспечение доступа к в иерархии объектов представлено на рис. 6.

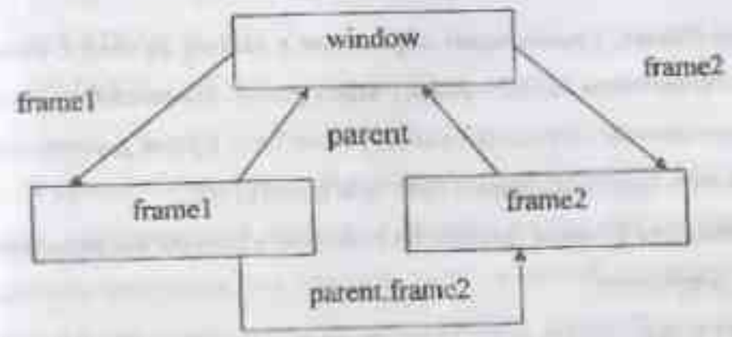


Рис. 6. Отношения в иерархии объектов

Так, при обращении из одного фрейма-потомка к другому необходимо помнить, что прямой связи между фреймами-потомками не существует. Поэтому сначала нужно обратиться к родительскому окну, а затем к его второму потомку:

parent. frame2. document. write(" Привет от первого фрейма ").

Можно изменить элемент одного фрейма из другого. Например, при щелчке на тексте в правом фрейме в левом изменится один из текстовых документов. Тогда документ в левом фрейме с именем LEFT:

```
<HTML>
Делай раз<BR>
Делай два
<H1 ID = "XXX"> Делай три </H1>
</HTML>
```

Документ в правом фрейме:

```
<HTML>
<SCRIPT>
function change( ) {
parent.LEFT.document.all.XXX. innerText = "Делай пять!!!"
}
```

```
</SCRIPT>
<H1 onclick = "change( )"> Щелкни здесь</H1>
</HTML>
```

В теле функции change( ) происходит обращение к левому фрейму с именем LEFT (задается в установочном HTML-файле) через parent. Изменение элемента происходит за счет присвоения значения свойству innerText. Кроме данного свойства, можно использовать outerText, innerHTML или outerHTML.

Важно, что изменения в одном фрейме по событию в другом происходят без перезагрузки HTML-документа.

Фреймы удобно использовать при создании навигационных панелей. В одном фрейме располагаются ссылки, а второй предназначен для отображения документов, вызываемых при активизации соответствующих ссылок.

Пример

```
// установочный файл frame.htm
<HTML>
<FRAMESET COLS = "25%, 75%">
<FRAME SRC = "menu.htm" NAME = "menu" >
<FRAME SRC = "start.htm" NAME = "main" >
</FRAMESET>
</HTML>
```

Здесь start.htm – документ, который первоначально показан во фрейме main.

```
//menu.htm – навигационная панель:
<HTML>
<SCRIPT>
function load (url) {
parent. main. location. href = url;
}
</SCRIPT>
<BODY>
```



```
<A HREF = "javascript:load('первый.htm')">Первый </A>
<A HREF = "второй.htm" TARGET = "main"> Второй </A>
<A HREF = "третий.htm" TARGET = "top"> Третий </A>
</BODY>
</HTML>
```

В примере окно браузера разделено на два фрейма. Первый из них исполняет роль навигационной панели, а второй – окна для отображения документов. Продемонстрированы два способа загрузки новой страницы во фрейм main. В первом случае используется функция load( ), параметр которой указывает, какой файл следует загрузить. При этом место, в которое он загружается, определяется самой функцией load( ). Во второй ссылке используется атрибут TARGET. В третьем ссылке демонстрируется, как можно избавиться от фреймов.

Для удаления фрейма с помощью load( ) достаточно записать:

```
parent. location. href = url
```

Атрибут TARGET в теге ссылки <A HREF> обычно применяется в случаях, когда требуется загрузить одну страницу в один фрейм. Язык сценариев используют при необходимости выполнения нескольких действий.

Для ссылок из родительского окна к объектам его дочерних фреймов можно использовать коллекцию frames. Обращение к определенному фрейму из этой коллекции возможно по индексу или по имени фрейма:

```
window. frames [индекс]
window. имя_фрейма
```

При обращении к объекту документа, загруженного во фрейм, следует сначала упомянуть объект document:

```
window. frames(0). document. all. MyInput. Value
window. LEFT. document. all. MyInput. Value
```

Ссылка из дочернего фрейма на родительский осуществляется с использованием parent.

При использовании top следует учитывать, что создаваемый сайт может

быть загружен в другой. Тогда объект top окажется объектом другого сайта. Поэтому лучше использовать parent для ссылок на вышестоящее окно или фрейм.

Ссылки top или self используют для предотвращения отображения сайта внутри фреймов другого сайта. Сценарий, выполняющий это, следует разместить в начале документа, например:

```
<SCRIPT>
if (top != self)
    top.Location = location
</SCRIPT>
```

Таким образом, ссылка на свойство top верхнего окна, должна совпадать со ссылкой self на текущее окно.

Для вставки одного HTML-документа в тело другого средствами браузера служит контейнерный тег <IFRAME>:

<IFRAME SRC = "адрес документа" > </IFRAME>

Данный элемент представляет собой прямоугольную область с прокруткой или без. Подобное окно называют плавающим фреймом. Такой документ можно позиционировать с помощью параметров таблицы стилей (тег <STYLE> или атрибут STYLE).

Плавающий фрейм аналогичен обычному фрейму. При создании он помещается в коллекцию frames. Среди его свойств широко используется align – выравнивание плавающего фрейма относительно окружающего содержимого документа. Его возможные значения:

absbotton – выравнивает нижнюю границу фрейма по подстрочной линии символов окружающего текста;

absmiddle – выравнивает середину границу фрейма по центральной линии между top и absbotton окружающего текста;

baseline – выравнивает нижнюю границу фрейма по базовой линии окружающего текста;

bottom – совпадает с baseline (только IE);

left – выравнивает фрейм по левому краю элемента-контейнера;

middle – выравнивает воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма;

right – выравнивает фрейм по правому краю элемента-контейнера;

textop – выравнивает верхнюю границу фрейма по надстрочной линии символов окружающего текста;

top – выравнивает верхнюю границу фрейма по верхней границе окружающего текста.

Задание

Вариант 1. Окно браузера поделить на два горизонтальных фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в левом фрейме по щелчку мыши на миниатюре изображения в правом фрейме. В этом фрейме организовать навигационную панель. Создать новый HTML-документ и вставить его в ранее созданный как плавающий вертикальный фрейм, выравнивая нижнюю границу фрейма по базовой линии окружающего текста.

Вариант 2. Окно браузера поделить на три горизонтальных фрейма. В левом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в среднем окне по щелчку мыши на миниатюре изображения в правом фрейме. В центральном фрейме организовать навигационную панель. Создать новый HTML-документ и вставить его в ранее созданный как плавающий вертикальный фрейм, выравнивая воображаемую центральную линию окружающего текста по воображаемой центральной линии фрейма.

Вариант 3. Окно браузера поделить на три вертикальных фрейма. В верхнем фрейме расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в среднем окне по щелчку мыши на миниатюре изображения в верхнем фрейме. В нижнем фрейме дополнительно организовать навигационную панель для среднего фрейма. Создать новый HTML-документ и вставить его в ранее созданный как плавающий фрейм, выравнивая

нижнюю границу фрейма по верхней границе текста.

Вариант 4. Окно браузера поделить на два фрейма. В нижнем расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в верхнем окне по щелчку мыши на миниатюре изображения в нижнем фрейме. Создать фрейм, горизонтальный, относительно двух ранее созданных. В нем организовать навигационную модель для верхнего фрейма. Создать новый HTML-документ и вставить его в ранее созданный как плавающий фрейм, выравнивая нижнюю границу фрейма по верхней границе текста.

Вариант 5. Окно браузера поделить на два фрейма. В верхнем расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в нижнем по щелчку мыши на миниатюре изображения, расположенного в верхнем фрейме. Создать фрейм, горизонтальный, относительно верхнего, из ранее созданных. В нем организовать навигационную модель для одного из ранее созданных фреймов. Создать два новых HTML-документа и вставить их в ранее созданный как плавающие фреймы, выравнивая их верхнюю границу по надстрочной линии символов окружающего текста.

Вариант 6. Окно браузера поделить на четыре горизонтальных фрейма. В правом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в самом левом окне по щелчку мыши на миниатюре изображения правого фрейма. Создать вертикальный фрейм, относительно ранее созданных. В нем организовать навигационную модель для второго фрейма. Создать новый HTML-документ и вставить его в ранее созданный документ как плавающий фрейм, выравнивая верхнюю границу фрейма по верхней границе окружающего текста.

Вариант 7. Окно браузера поделить на три горизонтальных фрейма. В правом расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в самом левом окне по щелчку мыши на миниатюре изображения правого фрейма. В центральном фрейме создать два вертикальных фрейма, в нижнем организовать навигационную модель для верхнего. Создать новый HTML-документ и вставить его в ранее созданный документ как

плавающий фрейм, выравнивая верхнюю границу фрейма по верхней границе окружающего текста.

Вариант 8. Окно браузера поделить на четыре равных фрейма. В правом нижнем расположить небольшое изображение. Организовать возможность вывода полномасштабного изображения в нижнем левом окне по щелчку мыши на миниатюре изображения правого фрейма. В верхнем левом фрейме организовать навигационную модель для верхнего правого фрейма. Создать новый HTML-документ и вставить его в ранее созданный документ как плавающий фрейм, выравнивая верхнюю границу фрейма по базовой линии текста.

Вариант 9. Окно браузера поделить на два горизонтальных фрейма. В правом – расположить четыре вертикальных фрейма, разместив в верхнем небольшое изображение. Организовать возможность вывода полномасштабного изображения в левом фрейме по щелчку мыши на миниатюре изображения. В нижнем из вертикальных организовать навигационную панель для вышележащего фрейма. Создать новый HTML-документ и вставить его в ранее созданный документ как плавающий фрейм, выравнивая его верхнюю границу по верхней границе окружающего текста.

Вариант 10. Окно браузера поделить на два вертикальных фрейма. В верхнем разместить небольшое изображение. В нижнем – три горизонтальных фрейма. Организовать возможность вывода по щелчку мыши на миниатюре изображения полномасштабного изображения в среднем нижнем. В правом организовать навигационную панель для левого фрейма. Создать новый HTML-документ и вставить его в ранее созданный документ как плавающий фрейм, выравнивая верхнюю границу фрейма по верхней границе окружающего текста.

Контрольные вопросы

1. Поясните понятие «фрейм».
2. Объясните механизм создания навигационной панели.
3. Какой тег служит для вставки одного HTML-документа в тело другого средствами браузера?

4. Перечислите различия между обычным фреймом и плавающим.

Лабораторная работа №7

Применение фильтров

С помощью фильтров каскадных таблиц стилей можно получить разнообразные эффекты: постепенное появление или исчезновение рисунка, плавное преобразование одного изображения в другое, задание степени прозрачности и др.

В сценариях графика сохраняется в файлах форматов gif, png, jpeg, swf.

GIF поддерживает чересстрочную загрузку, прозрачность пикселей и анимацию. Однако глубина цвета в нем ограничена 256, а пиксели не могут быть полупрозрачными.

PNG имеет очень много общего с форматом GIF, но позволяет сохранять многоцветное изображение, а анимацию не поддерживает.

JPEG позволяет сохранять полноцветные изображения фотографического качества и загружать их в чересстрочном режиме. Данный формат не поддерживает прозрачность и анимацию.

SNF сохраняет векторную графику и анимацию, созданные в пакете MacroMedia Flash, а также импортированные растровые изображения и звуковое сопровождение. Прозрачность изображений может принимать множество значений от 0 до 100.

Форматы gif и jpeg очень популярны в веб-дизайне.

Фильтр следует понимать как некий инструмент преобразования изображения, взятого из графического файла и вставленного в HTML- документ с помощью тега <IMG>.

Фильтры можно применять не только к графическим объектам, но и к текстам, текстовым областям, кнопкам.

С помощью фильтра alpha можно установить прозрачность графического объекта. Сквозь прозрачные графические объекты видны нижележащие изображения. Прозрачность имеет несколько вариантов градиентной формы. Например,

может увеличиваться от центра к краям изображения.

Фильтр alpha задается с помощью каскадной таблицы стилей и имеет ряд параметров. В примере для графического изображения стиль определяется с помощью атрибута STYLE:

```
<IMG ID = "myimg" SCR = " pict. gif "
STYLE = "position: absolute; top:10; left: 50;
filter: alpha (opacity = 70, style = 3) ">
```

Здесь целочисленный параметр opacity определяет степень непрозрачности. Значение 0 соответствует полной прозрачности изображения, а 100 – полной непрозрачности. Параметр style задает градиентную форму распределения прозрачности по изображению как целое число от 0 до 3. По умолчанию значение параметра равно 0, и градиент не применяется. Фильтр имеет и другие параметры, определяющие прямоугольную область изображения, к которой применяется фильтр. По умолчанию фильтр применяется ко всему изображению.

Фильтр можно определить в каскадной таблице стилей внутри контейнерного тега <STYLE>:

```
<HTML>
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: alpha (opacity = 70, style = 3)}
</STYLE>
< IMG ID = "myimg" SCR = " pict. gif "
</HTML>
```

Доступ к свойствам фильтра в сценарии:

document. all. id_изображения. filters ["имя_фильтра"]. параметр = значение

Для рассмотренного примера это выражение имеет вид:

document. all. myimg. filters ["alpha"]. opacity = 30

Для остальных параметров alpha аналогично.

Для IE5.5+ можно использовать другой синтаксис, в котором в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра:

#myimg { filter: progid: DXImageTransform . Microsoft . alpha (opacity = 70, style = 3)}

Тогда доступ к свойствам фильтра:

document.all.myimg.filters ["DXImageTransform. Microsoft alpha"]. opacity = 30

Фильтр alpha статический. Существуют и динамические фильтры: apply( ) – фиксирует изображение, play( ) его трансформирует, revealtrans( ) преобразовывает изображение, stop( ) при необходимости останавливает процесс преобразования.

Фильтр revealtrans( ) имеет параметры: duration – длительность преобразования в секундах (число с плавающей точкой) и transition – тип преобразования (целое от 0 до 23).

Для эффекта появления изображения можно воспользоваться фрагментом, который происходит после загрузки документа, т.е. по событию onload:

```
<HTML>
<BODY onload = "transform( )" >
<IMG ID = "myimg" SRC = "pict. gif " STYLE = "position: absolute; top:10;
left: 50; visibility = "hidden" filter: revealtrans (duration = 3, transition =12) " >
//transition =12 соответствует плавной трансформации
</BODY>
<SCRIPT>
function transform ( ) {                          //появление изображения
document.all.myimg.style. visibility = "hidden"   // изображение невидимо
myimg. filters ("revealtrans"). apply( )
myimg.style. visibility = "visible"
myimg. Filters ("revealtrans"). play( )           // выполняем преобразования
}
</SCRIPT>
</HTML>
```

Для замены одного изображения на другое необходимо установить начальное и конечное изображения путем присвоения нужных значений свойству src объекта, соответствующего изображению, – например фрагментом:

```
document.all.myimg.src = "pict2. gif"
```


Рассмотренный синтаксис воспринимается браузерами IE4+. Для IE5.5+ в каскадной таблице стилей задается ссылка на специальный компонент и имя фильтра. Так для трансформации изображения по щелчку мыши на графическом объекте в другое, и обратно, можно воспользоваться программой:

```
<HTML>
<STYLE>
#myimg{position: absolute; top:10; left: 50; filter: progid: DXImageTransform.
Microsoft revealtrans (duration = 3, transition = 12)}
</STYLE>
<IMG ID = "myimg" onclick = "transform( )" SCR = "ear. gif ">
<SCRIPT>
function transform( ) {
//фиксация исходного изображения
myimg. filters ("DXImageTransform . Microsoft revealtrans") apply ( )
//определение конечного изображения
if (document. all. myimg. src. index0f ("ear")!= -1)
document. all. myimg. src = "s gif"
      else document. all. myimg. src = "ear.gif"
//выполняем преобразование
myimg. filters ("DXImageTransform . Microsoft revealtrans"). play ( )
}
</SCRIPT>
</HTML>
```

В браузере IE5.5+ возможно применение фильтра basicimage, с помощью которого изображение можно повернуть на угол, кратный 90 градусам, задать прозрачность, зеркально отразить и др.

Параметр rotation фильтра basicimage принимает целочисленные значения: 0 (нет поворота), 1 (поворот на 90°), 2 (поворот на 180°), 3 (поворот на 270°).

К видимому элементу можно применить несколько различных фильтров одновременно. Например, сделать объект полупрозрачным с помощью фильтра al-

pha и трансформировать его с помощью фильтра revealtrans.

Задание

В сценарии предыдущей лабораторной работы к изображениям применить различные фильтры и их сочетания. При необходимости – добавить новые события.

Контрольные вопросы

1. Приведите характеристики различных форматов хранения графического изображения.

2. К какому из фильтров относятся параметры duration и transition? За какие виды преобразований они отвечают?

Лабораторная работа № 8

Движение элементов

Перемещение видимых элементов HTML-документа в окне браузера основано на изменении значений параметров позиционирования top и left таблицы стилей. Эти параметры можно указывать в атрибуте STYLE или в теге <STYLE> при задании видимых элементов.

Затем в сценарии определяется способ изменения параметров координат top и left. Можно заставить элемент перемещаться постоянно, в течение заданного времени или в ответ на события (например, по щелчку кнопкой мыши).

Оператор цикла для организации расчета координат обычно не применяется, поскольку пока выполняется цикл, другие выражения сценария не работают.

Схема сценария, осуществляющего непрерывное перемещение видимого элемента документа, имеет следующий вид:

```
function in_move( ){                    // инициализация движения
...                                     // подготовка к запуску функции move( )
setInterval ( "move( )", временная задержка )
function move( ){
...                    // изменение top и left стиля перемещаемого элемента
}
in_move( )                              // вызов функции для перемещения элемента
```


Метод setInterval имеет два параметра: строку, содержащую выражение, которое должно выполняться периодически через количество миллисекунд, заданное в качестве второго параметра. При этом координаты элемента меняются постоянно и создается эффект движения. Скорость и плавность движения зависят от величин приращения координат (в функции move( )) и временной задержки (второго параметра метода setInterval( )).

Движение по прямой линии можно реализовать следующим сценарием:

```
<HTML>
IMG ID="myimg" SRC="pict.gif" STYLE="position : absolute; top : 20; left : 15">
<SCRIPT>
function mmm( ) {
dx = 5                                  // приращение по y
dy = 7                                  // приращение по x
setInterval ("go ( ) ", 250)            // периодический вызов функции
go ( )
}
function go( ){                         // изменение координат изображения
/* Текущие координаты: */
var y = parseInt (document. all .myimg. style . top)
var x = parseInt (document. all .myimg. style . left)
/* Новые координаты: */
document. all .myimg. style . top = y + dy
document. all .myimg. style . left = x + dx
```

```
}
mmm( )                                  // начинаем движение
</SCRIPT>
</HTML>
```

Переменные dx и dy являются глобальными и доступны в функции mmm(). Значения параметров позиционирования в таблице стилей должны иметь строковый тип. Поэтому потребовалось применение функции parseInt( ) для приведения их к целому типу.

Если идентификатор перемещаемого объекта, приращения координат и временную задержку задать параметрами функции mmm( ), то получится универсальная функция линейного перемещения любого видимого элемента.

```
function mmm (xid,  dx,  dy)  {
var  prmstr = "'" + xid + "'," +dx + "," + dy     // строка параметров
prmstr = "go(" + prmstr + ")"
 setInterval(prmstr ,  250)                  // периодический вызов go( )
}
function go(xid,  dx,  dy){
y = parseInt(document.all[xid].style.top)
x = parseInt(document.all[xid].style.left)
document.all.myimg.style . top = y + dy
document.all.myimg.style.left = x + dx
}
```

```
mmm("myimg",  12,  4)                       // пример вызова функции
```

Методу setInterval( ) передать в явном виде параметры нельзя, поскольку первый из них – строка.

В этом случае элемент будет двигаться и за границами браузера. Для его остановки служит метод clearInterval( ), единственным параметром которого является целочисленный идентификатор, возвращаемый методом setInterval( ).

Значение, возвращаемое методом setInterval( ), можно сохранить в глобальной переменной, а затем использовать его в качестве параметра метода clearInterval( ) в теле функции, имя которой передается в качестве первого параметра методу setInterval( ). Например:

```
if (parseInt(document.all[xid].style.left) > 350)  clearInterval(id_move)
```

В этом случае движение остановится, как только горизонтальная координата элемента превысит 350 пикселей.

Движение по эллипсу задается несколькими параметрами – такими как большая и малая полуоси, положение центра и угол поворота относительно горизонтали, угловая скорость перемещения и др.

В примере функция ellipse() является функцией инициализации движения, m( ) — функция, передаваемая методу setInterval().

```
function ellipse(xid,  alpha,  a,  b,  omega,  x0,  y0,  ztime,dt)  {
/* Параметры:
xid – Id  движущегося объекта, строка;
alpha – угол  поворота эллипса, градусы;
x0 – x-координата  центра эллипса, пиксели;
y0 – y-координата центра эллипса, пиксели;
a – большая  полуось эллипса, пиксели;
b – малая  полуось эллипса, пиксели;
omega – угловая  скорость, град/с, знак задает направление вращения;
ztime – начальная  фаза, градусы;
d – временная  задержка, секунды.
*/
// проверка наличия параметров:
if (!alpha)  alpha = 0
if (!a)  a = 0
if (!b)  b = 0
if (!omega)  omega = 0
if (!x0)  x0 = 0
if (!y0)  y0 = 0
if (!ztime)  ztime = 0
```

```
if (!dt)  dt = 0

var t = -ztime                    /* чтобы начальное значение было 0,
                                  поскольку в m( ) уже есть приращение*/
setInterval("m(" + xid + "," + alpha + "," + a +"," + b + "," + omega + "," + x0 +
"," + y0 + "," + ztime + "," + dt + ")", ztime*1000)
/*многократный вызов m( ) с интервалом ztime, мс */
function m(xid, alpha, a, b, omega, x0, y0, ztime, dt) {
/* пересчет координат, вызывается из ellipse( ) */
t+= ztime
/* x,y – координаты в собственной системе координат */
var x = a*Math.cos((omega*t + dt)*Math.PI/180)
var y = b*Math.sin((omega*t + dt)*Math.PI/180)
var as = Math.sin(alpha*Math.PI/180)
var ac = Math.cos(alpha*Math.PI/180)
document.all[xid].style.top = -x*as + y*ac + y0
document.all[xid].style.left = x*ac + y*as + x0
}
```

Движение по произвольной кривой, заданной выражениями, которые описывают изменения вертикальной и горизонтальной координат элемента, рассмотрим на примере перемещения по синусоиде с амплитудой 50 пикселей и горизонтальной скоростью 10 пикселей в секунду. Начальные координаты графического объекта равны 100 и 50 пикселей по вертикали и горизонтали соответственно.

```
function curvemove(xid,  yexpr,  xexpr,  ztime,  namevar)  {
/* Движение по произвольной кривой
xid – id  движущегося объекта,  строка
yexpr – выражение для  вертикальной координаты
xexpr – выражение для  горизонтальной координаты
ztime – интервал  времени между вызовами функции m( ),  мс
namevar – имя  аргумента в  выражениях yexpr  и  xexpr */
```



```
if  (!xid)  return null
if  (!yexpr)  yexpr = "x"
if  (!xexpr)  xexpr = "x"
if  (!ztime)  ztime = 100                    // интервал времени,  мс
if  (!namevar)  namevar = "x"
eval(namevar  + "= 0")                  /*  глобальная  переменная,  входящая
                                        в выражения yexpr и xexpr  */
setInterval("m(" + xid + "',  '" + yexpr + "',  '" + xexpr + "',  '" + namevar +
"')",  ztime) }
function m (xid,  yexpr,  xexpr,  namevar)  {
eval(namevar  + "++")
document.all[xid].style.top = eval(yexpr)
document.all[xid] . style.left  = eval(xexpr)
}
```

Имя аргумента передается функциям в качестве строкового параметра namevar. Чтобы создать переменную с именем, указанным в значении переменной namevar и присвоить ей некоторой значение, необходимо создать строку с оператором присвоения и передать ее функции eval( ).

Тогда вызов функции может быть одним из следующих:

```
curvemove("myimg1",  "100 + 100*Math.sin(0.05*w)/(0.05*w)",  "50−w",  50,  "w")
curvemove("myimg2","100 + 2*time)",  "570 -  time",  100,"w")
```

Можно перемещать изображения мышью различными способами. Один из них в следующем: пользователь пытается перетащить мышью изображение, затем отпускает кнопку мыши и перемещает указатель в нужное место, где отпускает кнопку мыши или щелкает ею.

```
//Код перемещения изображения мышью
<HTML>
<HEAD><TITLE>перемещаемая картинка</TITLE></HEAD>
<B0DY id = "mybody">
<IMG ID="myimg" SRC = p.gif ondragstart = "drag( )" style =
```

```
"position : absolute; top:15; left:20">
 </B0DY>
<SCRIPT>
flag = false                                    // нельзя перемещать
var id_img = " "
function drag( ){
flag = !flag
id_img = event.srcElement.id
}
function mybody.onmousemove( ){
if (flag){                                      // если можно перемещать
document.all[id_img].style.top = event .clientY
document.all[id_img].style.left = event .clientX
}
}
function mybody . onmouseup( ) {
flag = false                                    // нельзя перемещать
}
</SCRIPT>
</HTML>
```

Функция drag( ), обрабатывающая событие ondragstart (попытка перетаскивания), устанавливает переменную-триггер flag и выясняет, кто инициатор события. Значение переменной flag позволяет определить, можно или нельзя перемещать элемент. В данном примере инициатором события может быть только один элемент.

Задание

Вариант 1. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по прямой и его остановкой в нижнем левом углу страницы.

Вариант 2. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по окружности и остановкой через 10 секунд.

Вариант 3. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по синусоиде и его остановкой в верхнем левом углу страницы.

Вариант 4. Дополнить задание предыдущей лабораторной работы перемещением увеличенного изображения мышью.

Вариант 5. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по прямой и его остановкой в нижнем правом углу страницы.

Вариант 6. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по синусоиде и его остановкой в верхнем левом углу страницы.

Вариант 7. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по синусоиде и его остановкой в верхнем правом углу страницы.

Вариант 8. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по спирали и его остановкой через пять секунд.

Вариант 9. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по произвольной кривой и его остановкой через 7 секунд.

Вариант 10. Дополнить задание предыдущей лабораторной работы движением увеличенного изображения по спирали и его остановкой в правом верхнем углу страницы.

Контрольные вопросы

1. Назовите параметры позиционирования таблицы стилей, в каких тегах возможно их использование?

2. Каким образом преобразовать функцию движения изображения по эллипсу в движение по окружности?

3. Почему при реализации движения нельзя применять оператор цикла для пересчета координат?

4. По какой причине параметры функций, отвечающих за перемещение, являются глобальными переменными?

Основы HTML

HTML – язык разметки гипертекста – является основным языком программирования web-страниц. Описания web-страниц содержатся в HTML-программах (HTML-кодах), которые хранятся в обычных текстовых файлах с расширением htm или html. Программы на языке HTML содержат инструкции, называемые тегами. Теги представляют собой последовательности символов, заключенные в угловые скобки < >.

Большинство современных браузеров допускает запись в любом регистре. Но чаще их пишут прописными буквами. Обычно ключевые слова тегов являются аббревиатурами английского языка.

Любая HTML-программа должна начинаться тегом <HTML> и заканчиваться тегом </HTML>. Некоторые теги используются только парами, при этом первый из них называется открывающим, а второй – закрывающим. Иногда парные теги называются контейнерными, поскольку между ними можно разместить и другие теги. Существуют также одиночные теги.

Теги могут содержать параметры, называемые атрибутами, могут иметь значения – аргументы.

Некоторым тегам соответствуют видимые элементы HTML-документа (заголовки, фрагменты текста, рисунки и т.п.), а некоторые выполняют специальные функции, не связанные с выводом в окно браузера. Так, все заключенное между символами <! > является комментарием и не отображается в окне браузера.

HTML-программа состоит из тела и заголовка. Каждая из этих частей ограничивается парными тегами: заголовок – <HEAD> </HEAD>, а тело – <BODY> </BODY>. Внутри тегов заголовка можно поместить теги <TITLE> </TITLE>, расположив в них текст, который появится в заголовке браузера. Таким образом, HTML-документ в общем случае имеет вид:

```
<HTML>
    <HEAD> <TITLE> Текст в заголовке </TITLE></HEAD>
```

<BODY>

Здесь могут располагаться теги, тексты, рисунки, ссылки на другие документы и др.

</BODY>

</HTML>

Писать каждый тег с новой строчки или делать отступы необязательно, однако это улучшает восприятие программы.

В HTML-программу можно вставить фрагменты, созданные на JavaScript, заключив их в контейнер <SCRIPT> </SCRIPT>.

Текст HTML-документа может быть выполнен в обычном окне текстового редактора Блокнот. Вместо него также можно использовать специальные редакторы web-страниц, – например, Microsoft FrontPage или Macromedia Dreamweaver.

Основные теги HTML

Назначение	Формат	Значения аргументов
Структура web-страницы		
Начало и конец страницы	<HTML> </HTML>	
Описание страницы, в том числе со имя	<HEAD> </HEAD>	
Имя страницы	<TITLE> </TITLE>	
Содержание страницы	<BODY> </BODY>	
Форматирование текста		
Заголовок (уровни от 1 до 6)	<H?> </H?>	
Заголовок с выравниванием	<H? ALIGN= '**'> </H?>	left center right
Абзац	<P> </P>	
Абзац с выравниванием	<P ALIGN="**"></P>	left center right
Перевод строки	 	
Горизонтальный разделитель	<HR>	
Выравнивание по центру	<CENTER> </CENTER>	
Адрес автора	<ADDRESS> </ADDRESS>	
Форматирование шрифта		
Жирный	<B> </B>	

Курсив	<I> </I>	
Верхний индекс		
Нижний индекс		
Размер шрифта (от 1 до 7)	<FONT SIZE=?> </FONT>	
Цвет шрифта (задается названием цвета или его 16-ричным кодом)	<FONT COLOR="#RRGGBB"> </FONT>	red blue #FFFFFF и др.
Гарнитура шрифта	<FONT FACE="*"> </FONT>	Arial TimesET и др.
Вставка изображений		
Вставка изображения	<IMG SRC="URL">	
Выравнивание текста около изображения	<IMG SRC="URL" ALIGN="*">	top bottom middle left right
Вывод текста вместо изображения	<IMG SRC="URL" ALT="*">	текст
Цвет фона, текста и ссылок		
Фоновое изображение	<BODY BACGROUND="URL">	
Цвет фона	<BODY BGCOLOR="#RRGGBB">	red blue #FFFFFF и др.
Цвет текста	<BODY TEXT="#RRGGBB">	
Цвет ссылки	<BODY LINK="#RRGGBB">	
Цвет пройденной ссылки	<BODY VLINK="#RRGGBB">	
Цвет активированной ссылки	<BODY ALINK="#RRGGBB">	

Вставка гиперссылок		
Ссылка на другую страницу	<A HREF="URL">указатель ссылки</A>	
Ссылка на закладку в другом документе	<A HREF="URL#*"> указатель ссылки </A>	
Ссылка на закладку в том же документе	<A HREF="#*">указатель ссылки</A>	
Определение закладки	<A NAME="*"> </A>	
Списки		
Неупорядоченный	<UL> <LI> </UL>	
Тип метки	<UL TYPE="*">	disk circle square
Нумерованный	<OL> <LI> </OL>	
Тип нумерации	<OL TYPE="*">	A, a, I, i, 1
Первый номер списка	<OL START=?>	1,2, ...
Список определений <DT>термин <DD>определение	<DL> <DT> <DD> </DL>	
Меню	<MENU> <LI> </MENU>	
Каталог	<DIR> <LI> </DIR>	
Формы		
Форма	<FORM> </FORM>	
Текстовое поле	<INPUT TYPE="text"	1,2,3 ...

NAME="name"	NAME="name" SIZE=?>	
Группа переключателей NAME="group"	<INPUT TYPE="radio" NAME="group" VALUE= "*">	rad1 rad2 rad3
Группа флажков NAME="group"	<INPUT TYPE="checkbox" NAME="group" VALUE="*">	ch1 ch2 ch3
Раскрывающийся список NAME="list"	<SELECT NAME="list"> <OPTION>Первый <OPTION>Второй </SELECT>	
Текстовая область NAME="resume"	<TEXTAREA> NAME="resume" ROWS=? COLS=?> </TEXTAREA>	1,2,3 ...
Кнопка *Отправить*	<INPUT TYPE="submit" VALUE="Отправить">	
Кнопка *Очистить*	<INPUT TYPE="reset" VALUE="Очистить">	

Свойства объекта window

parent	возвращает родительское окно для текущего;
self	возвращает ссылку на текущее окно;
top	возвращает ссылку на главное окно;
name	название окна;
opener	окно, создаваемое текущим;
closed	сообщает, если окно закрыто;
status	текст, показываемый в строке состояния браузера;
defaultStatus	текст по умолчанию строки состояния браузера;
returnValue	позволяет определить возвращаемое значение для события или диалогового окна;
client	ссылка, возвращаемая объект навигатора браузеру;
document	ссылка только для чтения на объект окна document;
event	ссылка только для чтения на глобальный объект event;
history	ссылка только для чтения на объект окна history.
location	ссылка только для чтения на объект окна location;
navigator	ссылка только для чтения на объект окна navigator;
screen	ссылка только для чтения на объект окна screen.

Методы объекта window

open( )	открывает новое окно браузера;
close( )	закрывает текущее окно браузера;
showHelp( )	показывает окно подсказки как диалоговое;
showModalDialog( )	показывает новое модальное(диалоговое) окно;
alert( )	окно предупреждения с сообщением и кнопкой OK;
prompt( )	окно приглашения с сообщение, текстовым полем и кнопками OK и Cancel (Отмена);

confirm( )	окно подтверждения с сообщением и кнопками OK и Cancel;
navigate( )	загружает другую страницу с указанным адресом;
blur( )	убирает фокус с текущей страницы;
focus( )	устанавливает страницу в фокус;
scroll( )	разворачивает окно на заданную ширину и высоту.
setInterval( )	указывает процедуре выполняться автоматически через заданное число миллисекунд;
setTimeout( )	запускает программу через заданное количество миллисекунд после загрузки страницы;
clearInterval( )	обнуляет таймер, заданный методом setInterval( );
clearTimeout( )	обнуляет таймер, заданный методом setTimeout( );
execScript( )	выполняет код сценария, по умолчанию Jscript.

События объекта window

onblur	выход окна из фокуса;
onfocus	окно становится активным;
onhelp	нажатие пользователем клавиши F1;
onresize	изменение пользователем размеров окна;
onscroll	прокрутка окна пользователем;
onerror	ошибка при передаче;
onbeforeunload	для сохранения данных перед выгрузкой страницы;
onload	страница полностью загружена;
onunload	непосредственно перед выгрузкой страницы.

Свойства объекта document

Свойство	Атрибут	Назначение
activeElement		Активизирует активный элемент
alinkColor	ALINK	Цвет ссылок на странице
bgColor	BGCOLOR	Определяет цвет фона элемента
body		Ссылка только для чтения на неявный основной объект документа, определенный в теге <BODY>
domain		Устанавливает или возвращает домен документа для его защиты или идентификации.
fgColor		Устанавливает цвет текста переднего плана
lastModified	TEXT	Дата последней модификации страницы, доступна как строка
linkColor	LINK	Цвет еще непосещенных гиперссылок на странице
location		Полный URL документа
parentWindow		Возвращает родительское окно для документа
readyState		Определяет текущее состояние загружаемого объекта
referer		URL страницы, которая вызвала текущую
selection		Ссылка только для чтения на дочерний для document объект selection
title	TITLE	Определяет справочную информацию элемента, используемую при загрузке и всплывающей подсказке
url	URL	URL-адрес документа клиента или в теге <META>
vlinkColor	VLINK	Цвет посещенных ссылок на странице

Методы объекта document

clear	очищает выделенный участок
close	закрывает текущее окно браузера
createElement	создает экземпляр элемента для выделенного тега
elementFromPoint	возвращает элемент с заданными координатами
execCommand	выполняет команду над выделенной областью
open	открывает документ
queryCommandEnabled	сообщает, доступна ли данная команда
queryCommandIndeterm	сообщает, если данная команда имеет неопределенный статус
queryCommandState	возвращает текущее состояние команды
queryCommandSupported	сообщает, поддерживается ли данная команда
queryCommandText	возвращает строку, с которой работает команда
queryCommandValue	возвращает значение команды, определенное для документа или объекта TextRange
write (writeln)	записывает текст и код HTML в документ, находящийся в указанном окне

События объекта document

onclick	при щелчке левой кнопкой мыши
ondblclick	при двойном щелчке левой кнопкой мыши
ondragstart	при возникновении перетаскивания
onerror	ошибка при передаче
onhelp	нажатие клавиши F1
onkeydown	нажатие клавиши
onkeypress	возникает при нажатии клавиши и продолжается при удержании клавиши в нажатом состоянии
onkeyup	пользователь отпускает клавишу

onload	при полной загрузке документа
onmousedown	при нажатии кнопки мыши
onmousemove	при перемещении указателя мыши
onmouseout	когда указатель мыши выходит за границы элемента
onmouseover	когда указатель мыши входит на документ
onmouseup	пользователь отпускает кнопку мыши
onreadystatechange	возникает при изменении свойства readystate
onselectstart	когда пользователем впервые запускается выделенная часть документа

Библиографический список

1. Дунаев В. Самоучитель JavaScript. – СПб.: Питер, 2005. – 395 с.

СОДЕРЖАНИЕ

Татьяна Алексеевна Галаган,

доц. каф. НиУС АмГУ, канд. техн. наук

Программирование на языке JavaScript. *Практикум*

Изд-во АмГУ. Подписано к печати 10.10.08. Формат 60x84/8. Усл. печ. л. 5,11.
Тираж 30. Заказ 240.
Отпечатано в типографии АмГУ